

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERÍA DE SISTEMAS

**DESARROLLO DE UNA APLICACIÓN PARA LA INTERACCIÓN
ENTRE UN GUANTE ELECTRÓNICO E IMÁGENES EN 3D DEL
CUERPO HUMANO Y UN CLÚSTER A TRAVÉS DE LA RED
AVANZADA PARA EL PROYECTO SISTEMA DE
ENTRENAMIENTO VIRTUAL PARA MEDICINA.**

**TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE
SISTEMAS**

Blanca Rocío Inaquiza Chile

Kléber Mauricio Jácome Gavilanes

DIRECTOR Ing. Washington Ramírez

Quito, febrero 2013

DECLARACIÓN

Blanca Rocío Inaquiza Chile y Kléber Mauricio Jácome Gavilanes, declaran bajo juramento que el trabajo aquí descrito es de propia autoría; que no ha sido previamente presentado para ningún proyecto de grado o calificación profesional; y que se ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración ceden los derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Blanca Rocío Inaquiza Chile

Kléber Mauricio Jácome Gavilanes

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Blanca Rocío Inaquiza Chile y Kléber Mauricio Jácome Gavilanes bajo mi dirección.

Ing. Washington Ramírez

Director de tesis

AGRADECIMIENTO

En este mundo tan cambiante e innovador que proporciona nuevas tecnologías para trabajar, pero con las dificultades que conlleva al enfrentar la incertidumbre, falta de información, desconocimiento que en general existe sobre estas tecnologías que son lanzadas al mercado para ser explotadas a su máxima capacidad. Por este motivo se requiere agradecer la ayuda tan desinteresada de aquellas personas que de una u otra manera colaboraron en el desarrollo de este proyecto.

Agradecimientos al Ing. Washington Ramírez tutor del proyecto, que ha contribuido con su asesoramiento teórico, predisposición en todo momento, colaboración facilitando material técnico y participación en la mejora del contenido del proyecto con sugerencias y consejos.

A todos ellos un saludo y gracias.

Blanca Rocío Inaquiza Chile

Kléber Mauricio Jácome Gavilanes

DEDICATORIA

Este proyecto va dedicado a mis padres especialmente a mi madre Concepción y hermanos Luis, Carlos, Patricio, Billy y Ramiro, cuyo apoyo y ánimo durante todos estos años de carrera universitaria han sido la mejor de las ayudas y motivaciones para seguir adelante.

Blanca Rocío Inaquiza Chile

DEDICATORIA

Este proyecto va dedicado a Carlos, Anita y Ricardo con mucho cariño, por su motivación, apoyo y por estar presentes en cada momento, especialmente a mi madre por su ayuda incondicional, le ofrezco mi esfuerzo y trabajo al realizar este proyecto.

Kléber Mauricio Jácome Gavilanes

RESUMEN

El presente proyecto tiene como finalidad la creación de una aplicación que simule la manipulación de diferentes capas que conforman al cuerpo humano utilizando herramientas 3D, con el objetivo de ofrecer una alternativa virtual moderna para el aprendizaje de la medicina. Esta simulación se apoyará en el uso de un guante electrónico que mediante el trabajo con eventos del teclado y mouse complementará el escenario virtual idóneo para adquirir los conocimientos necesarios al respecto.

Como primera instancia se contempla la creación y edición de texturas en formato *Joint Photographic Experts Groupe* y para las imágenes 3D con formato *Wavefront*. Luego se efectuará el desarrollo de una aplicación gráfica con el propósito de manipular el cuerpo humano en 3D, realizar procesos como: rotar, trasladar, zoom y perspectivas de visualización

La manipulación de imágenes del cuerpo humano se lo desarrollará por niveles o capas; de tal forma que pueda cumplir con la selección aislada de partes como: piel, músculos, órganos, sistemas y huesos.

El siguiente paso es la creación de un software que reciba datos del guante electrónico el cual emula los movimientos, acciones del mouse y eventos del teclado para la manipulación del aplicativo 3D, permitiendo un enlace entre ellos. Finalmente se describe como el programa 3D es ejecutado, evaluando varias opciones hasta encontrar la que cumple con los requerimientos del proyecto para ser paralelizado sobre un Clúster de visualización, para ello debe haber una configuración previa.

Posteriormente se mostrará la aplicación desarrollada en la cual se unen los conceptos explicados a lo largo del proyecto. Se definirá el método implementado, se presentarán las conclusiones y posibles aplicaciones que se pueden dar en un futuro.

ABSTRACT

This project has the aims at creating an application that simulates the handling of different layers that make up the human body using 3D tools, with the objective by to offer an alternative to virtual learning modern medicine. This simulation will be supported by the use of an electronic glove that by working with the keyboard and mouse events complement the virtual scenario one can acquire the necessary knowledge about it.

As first instance for the creation and editing textures in *Joint Photographic Experts Groupe* format and 3D images with *Wavefront* format. Then it made the development of a graphical application for the purpose of manipulating the human body in 3D, make processes such as rotate, translate, zoom and display perspectives.

The manipulation of images of the human body will develop by levels or layers, in such a way that it can meet the selection isolated parts as skin, muscles, organs, other system and bones.

The next step is to create software that receives data from the electronic glove which emulates the movements, mouse actions and keyboard events for handling 3D application, allowing a link between them. Finally is described as the 3D program executed, evaluating various options to find the one that meets the requirements of the project to be parallelized on a display Cluster, for it must have a previous configuration.

Then be displayed in the application developed which bind the concepts explained throughout the project. Define the method is implemented, will present the conclusions and possible applications that can be given in the future.

ÍNDICE

CAPÍTULO I	1
1. INTRODUCCIÓN	1
1.1 PLANTEAMIENTO DEL PROBLEMA	2
1.2 OBJETIVOS	2
1.2.1 General.....	2
1.2.2 Específicos	2
1.3 JUSTIFICACIÓN	3
1.4 ALCANCE	3
1.5 METODOLOGÍAS DE DESARROLLO DE SOFTWARE.....	4
1.5.1 Metodologías Tradicionales.....	5
1.5.2 Metodologías Ágiles	7
CAPÍTULO II	10
2. MARCO DE REFERENCIAS	10
2.1 ANTECEDENTES	10
2.2 CONCEPTUAL.....	11
2.3 MARCO TEÓRICO.....	12
2.3.1 Metodología XP (Programación Extrema)	12
2.3.2 Entorno para desarrollo 3D.....	15
2.3.3 Dispositivo Electrónico Iteración Entorno 3D.....	47
2.3.4 Clúster	48
CAPÍTULO III	55
3. SELECCIÓN DE HERRAMIENTAS PARA ENTORNO 3D.....	55
3.1 HTML	55
3.2 C++	57
3.3 Java.....	59

3.4	Biblioteca Gráfica Java 3D	65
3.5	Biblioteca Gráfica OpenSceneGraph	68
3.6	Entorno de desarrollo integrado Eclipse.....	71
3.7	3D Max Studio.....	73
3.8	Adobe Photoshop.....	76
3.9	Clúster Rocks.....	79
3.10	PROCESO DE DESARROLLO.....	83
3.10.1	Roles definidos en la metodología XP	83
3.10.2	Planificación	83
CAPÍTULO IV		248
4.	FUNCIONAMIENTO DEL SISTEMA.....	248
4.1	Prototipo 1 Implementación del Aplicativo con Java 3D en el Clúster.....	248
4.2	Prototipo 2. Implementación del aplicativo con OpenSceneGraph ..	250
4.3	Prototipo 3. Implementación del aplicativo con CGLX.....	251
CONCLUSIONES.....		252
RECOMENDACIONES		254
BIBLIOGRAFÍA		255
5.	ANEXOS	261
5.1	ANEXO A – HU1: CREACIÓN Y EDICIÓN DE IMÁGENES EN 3D..	261
5.1.1	Instalación de 3D Max Studio 7	261
5.1.2	Herramientas de 3D Max Studio 7	265
5.1.3	Añadir Texturas a objetos 3D	270
5.1.4	Fusión Diseño 3D-Texturas	271
5.2	ANEXO B – HU2: CREACIÓN DE UN APLICATIVO PARA MANIPULAR IMÁGENES 3D	273
5.2.1	Aplicativo con Java 3D	273

5.3 ANEXO C – HU3: CREACIÓN DE UNA PÁGINA WEB PARA MANIPULACIÓN DEL APLET DESARROLLADO	289
5.3.1 Visualización del Applet con HTML	289
5.4 ANEXO D – HU4: MANIPULACIÓN DEL DISPOSITIVO ELECTRÓNICO.....	291
5.4.1 Configuraciones C++	291
5.5 ANEXO E – HU5: APLICATIVO DESARROLLADO INCLUIDO EN EL CLÚSTER.....	295
5.5.1 Configuración e Instalación de Rocks Clúster	295
5.5.2 Aplicativo con OpenSceneGraph.....	304

ÍNDICE FIGURAS

CAPÍTULOS

Figura 2-1	Arquitectura HTML – Applet.....	18
Figura 2-2	Arquitectura XML	20
Figura 2-3	Arquitectura de aplicación host que incrusta aplicaciones Python	23
Figura 2-4	Arquitectura C++	24
Figura 2-5	Arquitectura Java	25
Figura 2-6	Transformación para la máquina virtual	26
Figura 2-7	Esquema OpenGL	27
Figura 2-8	Arquitectura Open Inventor	30
Figura 2-9	Arquitectura de Visualización Pipeline(Oleoducto) VTK.....	32
Figura2-10	Arquitectura OSG	33
Figura 2-11	Arquitectura Java 3D.....	34
Figura 2-12	Arquitectura CGLX	36
Figura 2-13	Arquitectura X11	37
Figura 2-14	Arquitectura de Visual Studio	38
Figura 2-15	Arquitectura Eclipse	40
Figura 2-16	Arquitectura Code::Blocks.....	41
Figura 2-17	Dispositivo – Entorno 3D.....	48
Figura 2-18	Arquitectura de Clúster HA-OSCAR.....	51
Figura 2-19	Esquema general de configuración física de un Clúster	53
Figura 2-20	Arquitectura de un Clúster tipo SSI[58]	54
Figura 3-1	Arquitectura HTML – Dispositivo Electrónico – Aplicativo 3D	56
Figura 3-2	Coordenadas del mundo real y secuencia de transformación de vértices	63

Figura 3-3	Coordenada dextrógiro	64
Figura 3-4	Coordenadas del mundo real y secuencia de transformación de vértices.....	64
Figura 3-5	Arquitectura Java 3D.....	66
Figura 3-6	Applet de Hola Mundo en Java 3D.....	68
Figura3-7	Ventana de compilación del “Hola Mundo” con OSG.....	71
Figura 3-8	Interfaz gráfica de 3D Max Studio	76
Figura 3-9	Sistema de un Clúster.....	79
Figura 3-10	Arquitectura OpenMPI.....	82
Figura 3-11	Ventanas para acoplar y editar las imágenes 3D.....	90
Figura 3-12	Ventana para convertir a puntos de vértice.....	91
Figura 3-13	Ventana de selección de geometrías Stándard Primitives	92
Figura 3-14	Ventana de selección de geometría Extended Primitives	92
Figura 3-15	Ventana de selección de geometría “Spring”	93
Figura 3-16	Ventana de modificaciones con UNWRAP UVW y textura en Photoshop	93
Figura 3-17	Ventana de textura asignado a la imagen.....	94
Figura 3-18	Ventana de acoplado de imagen editada.....	95
Figura 3-19	Ventana de órgano detallando y finalizando su construcción	96
Figura 3-20	Ventana con textura de órganos	96
Figura 3-21	Ventana con texturas editadas.....	97
Figura 3-22	Ventana de órganos final junto al esqueleto	97
Figura 3-23	Ventana de error en Java al cargar texturas	98
Figura 3-24	Símbolos que Representan Objetos en un Escenario Gráfico ...	99
Figura 3-25	Escenario gráfico para el Applet 3D.....	102
Figura 3-26	Diagrama casos de uso general del sistema	104
Figura 3-27	Diseño de Clases Java	111

Figura 3-28	Ventana de la capa de piel.....	116
Figura 3-29	Ventana de la capa músculos	117
Figura 3-30	Ventana de la capa órganos	118
Figura 3-31	Ventana de la capa sistema respiratorio	120
Figura 3-32	Ventana de la capa esqueleto.....	121
Figura 3-33	Ventana de la capa de selección de objetos.....	122
Figura 3-34	Ventana de la capa de eliminar objetos	125
Figura 3-35	Ventana de Applet con cuerpo humano capa piel, capa músculos, capa órganos, capa sistema respiratorio y capa esqueleto.....	139
Figura 3-36	Plantilla para editar	143
Figura 3-37	Plantilla HTML.....	143
Figura 3-38	Plantilla elaborada.....	144
Figura 3-39	Página “humano3d.html” con el Applet3D incluido.....	145
Figura 3-40	Página de inicio	146
Figura 3-41	Página de información	147
Figura 3-42	Página de visualización del Applet 3D	147
Figura 3-43	Fases para programación interacción Aplicativo – Dispositivo electrónico	152
Figura 3-44	Diseño de clases guante electrónico de datos	155
Figura 3-45	Eventos asociados al dispositivo electrónico	167
Figura 3-46	Dispositivo electrónico activado y controlando el aplicativo	172
Figura 3-47	Monitoreo del Clúster con Ganglia	176
Figura 3-48	VirtualHost en Apache[76]	178
Figura 3-49	Instalación de Rocks Clúster.....	179
Figura 3-50	Applet incluido dentro del Clúster.....	182
Figura 3-51	Estructura para el sistema del aplicativo con OpenSceneGraph	186

Figura 3-52	Diagrama casos de uso general del sistema	187
Figura 3-53	Diagrama casos de uso general del sistema	188
Figura 3-54	Diseño de Clases OSG.	192
Figura 3-55	Ventana de la capa de piel.....	194
Figura 3-56	Ventana de la capa músculos	195
Figura 3-57	Ventana de la capa órganos	197
Figura 3-58	Ventana de la capa sistema respiratorio	199
Figura 3-59	Ventana de la capa esqueleto.....	201
Figura 3-60	Ventana de la capa de selección de objetos.....	208
Figura 3-61	Ventana de la capa de eliminar objetos	210
Figura 3-62	Ventana de aplicativo con cuerpo humano capas piel, músculos, órganos, sistema respiratorio y huesos.....	211
Figura 3-63	Archivo “MTL” (.mtl) guardado desde 3D Max Studio del archivo de imagen Model_Muscular_Zygomaticus_Minor	212
Figura 3-64	Diseño de un modelo paralelo.....	214
Figura 3-65	Ventana de visualización con MPI en el FrontEnd y en los nodos	218
Figura 3-66	Diseño de pantalla de tiled[78].....	221
Figura 3-67	Barra de Herramientas y Grupo de servidores.....	221
Figura 3-68	Diagrama casos de uso general del sistema con CGLX	222
Figura 3-69	Diseño de Clases OSG.	226
Figura 3-70	Autenticación de licencia.....	227
Figura 3-71	IP del nodo Head	228
Figura 3-72	Nombre del nodo y Cuadro de diálogo con la tabla para crear el diseño de pantalla para los nodos.....	229
Figura 3-73	Vista 3D de los monitores agregados	229
Figura 3-74	Vista 3D de los monitores agregados	230

Figura 3-75	Ventana de la capa piel.....	230
Figura 3-76	Imagen .osg visualizada en el FrontEnd y en los nodos junto al dispositivo electrónico	240
Figura 3-77	Ventana de aplicativo con CGLX del cuerpo humano	241
Figura 3-78	Mensaje ocurrido al intentar actualizar la tarjeta gráfica Nvidia	242
Figura 3-79	Archivo “MTL” (.mtl) del archivo de imagen Model_Skin/cara.mtl	242
Figura 4-1	Página de visualización del Applet 3D en la Web	248
Figura 4-2	Ventana de iteración del dispositivo electrónico con el Applet 3D	249
Figura 4-3	Ventana de iteración del dispositivo electrónico con el aplicativo OSG	250
Figura 4-4	Ventana de objeto seleccionado con el dispositivo electrónico..	250
Figura 4-5	Aplicación con CGLX en el FrontEnd del Clúster y en los nodos de visualización	251
Figura 4-6	Aplicación con CGLX en el FrontEnd del Clúster y en los nodos de visualización	251

Anexo A

Figura A - 1	Configuración del paquete “tff”	261
Figura A - 2	Configuración del WINE	262
Figura A - 3	Información de usuario de 3D max7 y número serial.....	263
Figura A - 4	Ventana de ingreso a 3D max7	264
Figura A - 5	Ventana de inicialización para ejecutar o activar el producto .	264
Figura A - 6	Ejecución correcta del funcionamiento de 3D max7	265
Figura A - 7	Barras de Menú Autodesk 3D Max	265
Figura A - 8	Barras de Herramientas Autodesk 3D Max	266

Figura A - 9	Barras de Herramientas 2 Autodesk 3D Max	266
Figura A - 10	Panel de solapas Autodesk 3D Max	267
Figura A - 11	Ventana de Mapping	270
Figura A - 12	Ventana de edición de la imagen en Adobe Photoshop	271
Figura A - 13	Ventana de selección del archivo a abrir.	271
Figura A - 14	Ventana para elegir la textura	272

Anexo B

Figura B - 1	Configuración del JRE	274
Figura B - 2	Configuración de JDK	276
Figura B - 3	Actualización del Javac	277
Figura B - 4	Ejecución del archivo binario en Ubuntu 11.04	278
Figura B - 5	Ejecución del archivo binario en CentOS	278
Figura B - 6	Error de no encontrar la ruta de la librería 'libj3dcore-ogl.so' .	279
Figura B - 7	Archivos ubicados en la ruta específica dentro de 'i386' y dentro del ext.....	279
Figura B - 8	Archivos ubicados en la ruta especificada en Mozilla.....	280
Figura B - 9	Instalación de Eclipse	281
Figura B - 10	Ventana de instalación de Java 3DS File Loader	281
Figura B - 11	Ventana de instalación de Java 3DS File Loader en CentOS	283
Figura B - 12	Ventana para ubicación del archivo	283
Figura B - 13	Ventana para añadir Librerías	284
Figura B - 14	Ventana para añadir librerías ".jar"	284
Figura B - 15	Ventana de librerías ".jar" agregadas	285
Figura B - 16	Método para importar el proyecto	285
Figura B - 17	Ventana de búsqueda del archivo seleccionado.....	286

Figura B - 18	Selección de los archivo a importar	286
Figura B - 19	Cambio de Argumento en Eclipse	287
Figura B - 20	Diagrama General de clases Java 3D	288

Anexo D

Figura D - 1	Diagrama General de clases Java 3D	294
--------------	--	-----

Anexo E

Figura E - 1	Error de conexión	299
Figura E - 2	Vmware	300
Figura E - 3	Nueva Conexión VMware	300
Figura E - 4	Tamaño de disco VMware	301
Figura E - 5	Ventana de edit settings	301
Figura E - 6	Selección de opción “tile”	302
Figura E - 7	Ventana de librería a comprobar	305
Figura E - 8	Ventana de error de librería gráfica	305
Figura E - 9	Ventana de librería libGL.so.1.2 a comprobar localización	306
Figura E - 10	Ventana de librería Nvidia a desinstalar	306
Figura E - 11	Ventana de instalación de OpenSceneGraph.....	308
Figura E - 12	Creación de una mano con OSG utilizando geometrías.....	311
Figura E - 13	Diagrama de Clases OSG.	312
Figura E - 14	Archivo .pro.....	316
Figura E - 15	Diagrama de Clases CGLX.....	317

ÍNDICE TABLAS

Tabla 1-1	Metodologías Ágiles vs Tradicionales.	5
Tabla 2-1	Cuadro comparativo entre los diferentes lenguajes de marcado .	17
Tabla 2-2	Cuadro Comparativo – Características de Lenguajes de Programación.....	22
Tabla 2-3	Bibliotecas OpenGL y plataformas soportadas	29
Tabla 2-4	Cuadro Comparativo – Características de Lenguajes de Programación.....	38
Tabla 2-5	Explicación de aplicaciones informáticas de modelado y animación 3D.....	42
Tabla 2-6	Características de los clúster OSCAR, ROCKS y OpenMosix	50
Tabla 3-1	Cuadro de los atributos que un <i>tag</i> utiliza	57
Tabla 3-2	Formatos de archivos soportados por OSG.....	70
Tabla 3-3	Formatos de archivo soportados por 3D Max Studio	74
Tabla 3-4	Formatos de archivo soportado en Photoshop.....	78
Tabla 3-5	Historia de Usuario HU1.....	84
Tabla 3-6	Historia de Usuario HU2.....	85
Tabla 3-7	Historia de Usuario HU3.....	85
Tabla 3-8	Historia de Usuario HU4.....	86
Tabla 3-9	Historia de Usuario HU5.....	86
Tabla 3-10	Estimación de Historia de Usuario.	87
Tabla 3-11	Diseño de Clases.	111
Tabla 3-12	Librerías J3D para visualizar un objeto, su color y coordenadas	112
Tabla 3-13	Librerías J3D para cargar archivos .obj.....	113
Tabla 3-14	Librerías J3D para comportamiento personalizado	113
Tabla 3-15	Lista de capacidades de TransformGroup	114

Tabla 3-16	Lista de capacidades de otros aspectos del TransformGroup ..	115
Tabla 3-17	Eventos del teclado para manipulación de los objetos 3D	133
Tabla 3-18	Librerías C++.....	155
Tabla 3-19	Librerías Gráficas	156
Tabla 3-20	Librerías puerto USB.....	156
Tabla 3-21	Librerías propias.....	156
Tabla 3-22	Configuración asociada a los eventos del mouse	171
Tabla 3-23	Lista de clases OSG.....	191
Tabla 3-24	Librerías OSG	193
Tabla 3-25	Librerías OSG	193
Tabla 3-26	Eventos del teclado para manipulación de los objetos 3D	206
Tabla 3-27	Lista de clases CGLX.....	223
Tabla 3-28	Eventos del teclado para manipulación de los objetos 3D	236
Tabla 3-29	Distribución del trabajo en los nodos de visualización	245
Tabla 3-30	Distribución del trabajo en los nodos de visualización	246
Tabla 3-31	Distribución del trabajo en los nodos de visualización	247

CAPÍTULO I

1. INTRODUCCIÓN

Uno de los propósitos del presente trabajo de tesis es la creación de una herramienta de software, útil para la simulación de imágenes en 3D del cuerpo humano. Durante el desarrollo de este proyecto se destaca la labor de investigación que se ha realizado mediante un largo tiempo, buscando la mejor forma de abordar el proyecto con la ayuda de las nuevas tecnologías de información. El proyecto está dividido en secciones que describen diferentes aspectos necesarios para comprender el correcto funcionamiento del sistema.

El primer capítulo describe el problema, los objetivos que se pretenden alcanzar en el transcurso del mismo, se justifica el por qué se necesita implementarlo, sus alcances y la metodología a utilizar.

El segundo capítulo trata del marco de referencias, donde se describen y analizan las herramientas de software útil para el desarrollo de gráficos en tercera dimensión que modelen la estructura física del cuerpo humano en imágenes 3D y la comunicación con el dispositivo electrónico, el cual efectuará el envío de datos desde sus sensores para su posterior control e interacción.

El tercer capítulo hace alusión al marco metodológico que es donde se seleccionarán y configurarán las herramientas a utilizar para la manipulación, también se definirá el proceso de desarrollo tanto el análisis, diseño e implementación de software a seguir para la construcción del aplicativo.

En el cuarto capítulo se realizan las pruebas finales para verificar el funcionamiento del aplicativo, así como también la transferencia de datos a través del clúster y la red avanzada.

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad debido a los avances tecnológicos, la medicina requiere de ciertos adelantos que impliquen programas computacionales en línea, que sean óptimos, rápidos y que manejen datos en tiempo real, para solucionar los problemas que se presentan.

El proyecto “Sistema de Entrenamiento Virtual para Medicina” requiere de un aplicativo para enlazar, controlar y manipular los módulos entre los datos que un dispositivo electrónico¹ envíe y las imágenes virtuales en 3D del cuerpo humano, las cuales se encuentran distribuidas en un Clúster y potenciadas a través de la Red Avanzada. Además el aplicativo requiere el envío de información de los procesos realizados a un módulo “HTML” para interactuar con las imágenes virtuales por parte de los usuarios.

El dispositivo electrónico es un conjunto de sensores de movimiento y de contacto que enviarán datos en forma de cadena de caracteres a esta aplicación, éstos deberán ser usados para la interacción con las imágenes 3D.

1.2 OBJETIVOS

1.2.1 General

Desarrollar una aplicación que acople las señales que lleguen de los sensores de un guante, con secciones de imágenes digitales del cuerpo humano en 3D, desde un Clúster para el proyecto “Sistema de Entrenamiento Virtual para Medicina”.

1.2.2 Específicos

- Investigar y realizar el análisis de las librerías adecuadas para el manejo de gráficos en 3D para su funcionamiento y para la comunicación con un guante electrónico vía Wireless, el cual estará desarrollado en código abierto.

¹Guante electrónico

- Diseñar un módulo para el proyecto “Sistema de Entrenamiento Virtual para Medicina” que interactúe un “Guante Electrónico” con imágenes 3D de forma automática utilizando lenguajes de programación libre como: C++, Java, HTML.
- Implementar la paralelización de gráficos 3D del cuerpo humano, mediante los diferentes nodos del clúster a través de las redes avanzada y comercial, utilizando aplicaciones distribuidas.

1.3 JUSTIFICACIÓN

El desarrollo de este trabajo responde a la necesidad de implementar una aplicación para el proyecto “Sistema de Entrenamiento Virtual para Medicina” que contribuya a la finalización exitosa del mismo y que cumpla con los requerimientos específicos del proyecto aportando con los conocimientos adquiridos a lo largo de la carrera.

1.4 ALCANCE

El “Sistema de Entrenamiento Virtual para Medicina” requiere el desarrollo de un módulo que permita manipular mediante un dispositivo electrónico imágenes en 3D.

Las imágenes en 3D del cuerpo humano por medio de valores referentes a las coordenadas X, Y entregadas por un dispositivo electrónico en forma de cadena de caracteres, tendrán la capacidad de rotación, desplazamiento y escalado.

El dispositivo electrónico emulara los movimientos y acciones del mouse; los valores de las coordenadas serán acopladas a las imágenes del cuerpo humano para luego ser manipuladas en capas, empezando desde la piel hasta el sistema óseo, al cual tomaremos como base para la estructuración de los tejidos, órganos, sistemas y aparatos.

Se diseñará el sistema óseo así como también la epidermis en 3D. El referente será el sistema óseo con lo cual los tejidos, órganos, sistemas y aparatos tendrán un control que servirá para manipular y desglosar las imágenes desde la epidermis hasta el esqueleto humano, esto posibilita realizar transformaciones geométricas.

En conclusión, el proyecto se limita al desarrollo de una aplicación a manera de plugin para el “Sistema de Entrenamiento Virtual para Medicina”, la cual comunicará distintos módulos del proyecto.

La aplicación 3D para la comunicación del “Sistema de Entrenamiento Virtual para Medicina” servirá de base para el enlace de datos que vengan desde el Clúster al dispositivo electrónico.

Esta aplicación se podrá adaptar a los cambios que el sistema tendrá y puede haber opciones a expansión como por ejemplo la comunicación con otra aplicación como una página Web, la integración de más sensores en el dispositivo electrónico provocará el incremento de más datos. El nivel de flexibilidad de la aplicación irá de acuerdo a los cambios que se adopten en el transcurso del desarrollo del proyecto “Sistema de Entrenamiento Virtual para Medicina”.

Este proyecto no contempla el desarrollo y edición de las imágenes en 3D de los niveles intermedios como son tejidos, órganos, sistemas y aparatos, pero si el sistema óseo y epidermis. La aplicación se comunicará únicamente vía inalámbrica con el guante electrónico, el cual estará encargado de sensar la posición de la mano derecha del usuario y enviar los datos procesados en forma de cadena de caracteres para mover una imagen.

La aplicación no contempla la construcción del Clúster, solo la programación en paralelo para los diferentes nodos del Clúster y la aplicación por donde pasarán las imágenes en 3D del cuerpo humano.

No contempla el análisis, diseño, ni implementación del dispositivo electrónico, sino su uso para realizar pruebas de comunicación con la aplicación.

1.5 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software[1] se clasifican en dos grupos, las tradicionales y las ágiles.

Cuadro Comparativo

Metodologías Ágiles vs Tradicionales		
Características	Ágiles	Tradicionales
Cambios durante el proyecto	SI	NO
Valora a las personas y su iteración con el software, pasando por alto los procesos y herramientas para el desarrollo del mismo	SI	NO
Procesos mucho más controlados, con numerosas políticas/normas	NO	SI
No es muy flexible o tiene contrato tradicional	NO	SI
Software funcional a cambio de una documentación objetiva y exhaustiva	SI	NO
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	SI	NO
Pocos equipos (artefactos necesarios para elaborar el proyecto)	SI	NO
Más roles (mayores cargos a desempeñarse en el proyecto)	NO	SI
Mayor énfasis en la arquitectura de software	NO	SI
Comunicación entre desarrolladores y los clientes	SI	NO
Respuesta del software al cambio, por encima del seguimiento del plan elaborado	SI	NO
Simplifica el diseño para agilizar el desarrollo	SI	NO
Reciclado continuo de código (depuración)	SI	NO

Tabla 1-1 Metodologías Ágiles vs Tradicionales.

Fuente: Autores

Conociendo los aspectos de las metodologías se escogerá la mejor y la que más se acople al Proyecto, para ello se realiza un breve análisis de algunas de las principales.

1.5.1 Metodologías Tradicionales

Las metodologías tradicionales[2] en el desarrollo de software son elaboradas en etapas y de manera secuencial, tales como la RUP y la FLIP.

1.5.1.1 Proceso Unificado Racional (RUP)

RUP[3] tiene como objetivo entregar un producto. Es un proceso secuencial para el desarrollo de software que conjuntamente con el Lenguaje Unificado de Modelado UML, constituyen la metodología más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos para atender las necesidades específicas del proyecto.

A. Fases de RUP[4]

- **Inicio:** Su propósito es definir y acordar el alcance del proyecto con los patrocinadores, se identifican los principales casos de uso y los riesgos, se establece una visión general de la arquitectura de software.
- **Elaboración:** Se seleccionan los casos de uso adecuados para realizar la construcción de un plan de proyecto, se definen los mismo y se diseña una solución preliminar al problema
- **Construcción:** Se realizan tantas iteraciones sean necesarias hasta completar el funcionamiento del sistema con los cambios y mejoras haciéndolo eficiente con su respectivo manual de usuario.
- **Transición:** Se ajustan los errores y defectos encontrados en las pruebas de aprobación, verificando las especificaciones del producto y entregando al usuario el proyecto.

B. Etapas:

- Establece oportunidad y alcance
- Identifica las entidades externas o actores con las que se trata
- Identifica los casos de uso
- Procesos:
- Modelado de negocio
- Requisitos
- Análisis y Diseño
- Implementación

- Pruebas
- Despliegue

1.5.1.2 FuseboxLifecycleProcess (FLIP)

La metodología FLIP[5] ayuda a administrar el proceso de desarrollo de software de aplicaciones Web.

A. Fases de FLIP[6]

- **Personas y Objetivos:** Al conocer quién y para que se usará el software, se podrá entregar la aplicación que realmente el usuario necesita.
- **Wireframe:** Modela gráficamente lo que la aplicación realizará, presentando una idea rápida de lo que hará el sistema.
- **Prototipo y Diseño de interfaz de usuario:** Aprecia el plazo y el costo en la construcción del software, presenta la aplicación con el diseño final a entregar al cliente pero sin lógica programable.
- **Diseño de Aplicación:** Con el prototipo terminado se realiza el esquema de la aplicación organizándolo en circuitos.
- **Programación de código (Fusecoding):** El programador escribe el código para sus fusibles, los pone a prueba y acopla a la aplicación.
- **Pruebas Unitarias:** Se tiene un sistema de control de calidad para verificar el funcionamiento de los fusibles.
- **Integración de Aplicaciones:** Al finalizar cada circuito se los integra a la aplicación final.

1.5.2 Metodologías Ágiles

Las metodologías de desarrollo ágil[7] mejoran el desarrollo de sistemas y están indicadas en proyectos poco definidos y cambiantes mejorando las prácticas de desarrollo de software.

Las metodologías ágiles[8] muestran una ayuda para la superación de debilidades reales que se encuentran presentes en la Ingeniería del Software tradicional.

1.5.2.1 Programación Extrema (XP)

XP[3] se fundamenta en el trabajo orientado directamente al objetivo, basándose para esto en las relaciones interpersonales y en la velocidad de reacción para la implementación y para los cambios que puedan surgir durante el desarrollo del proceso.

La metodología XP se fundamenta en el trabajo en equipo, pone énfasis en la adaptabilidad a los cambios de requisitos en cualquier punto de la vida del proyecto empezando a valorar a los individuos e iteraciones por sobre procesos y herramientas, es decir, software funcionando por sobre documentación completa.

A. Fases de XP[8]

- Planificación del proyecto
- Diseño
- Codificación
- Pruebas

Estas fases se describirán con más detalle en el siguiente capítulo.

1.5.2.2 Desarrollo De Software Adaptable (ASD)

La ASD[9] no tiene muchas ataduras y reglas a seguir siendo la metodología más abierta. En ella no hay un ciclo de planificación-diseño-construcción del software, sino un ciclo especular-colaborar-aprender.

Esta metodología se basa en la adaptación continua a cambios, puesto que las necesidades del cliente varían regularmente.

A. Fases de ASD

- **Especulación:** Se establecerán los objetivos y las limitaciones con las que trabajará el proyecto.
- **Colaboración:** Es una fase cíclica, ya que lo aprendido para la construcción y gestión del producto por un equipo se transmitirá a los demás equipos.

- **Aprendizaje:** Se examina la calidad y funcionalidad del producto para luego ser entregado al cliente.

Una vez analizadas las metodologías descritas anteriormente se ha llegado a la conclusión que se aplicará la Programación Extrema por basarse en la simplicidad, la comunicación y el reciclado continuo de código.

CAPÍTULO II

2. MARCO DE REFERENCIAS

2.1 ANTECEDENTES

El hombre constantemente ha tenido la idea de modelar su entorno de la manera más real posible, por lo cual ha recreado una realidad virtual con diversas aplicaciones entre las cuales se encuentra la simulación virtual. En medicina, una de sus aplicaciones es permitir que futuros médicos practiquen varias veces en estos ambientes antes de entrar al quirófano, facilitando la elaboración de un diagnóstico del paciente o la simulación de una operación, también se han desarrollado proyectos como el de la universidad de Talca que ha combinado realidad virtual con equilibrio con el fin de estimular el aprendizaje motor en niños con parálisis cerebral [10] otros como los desarrollados por la universidad de Jaume para el tratamiento de fobias y trastornos alimenticios [11] por mencionar.

Se ha potenciado el trabajo con Clústers de visualización debido al bajo costo y a las grandes prestaciones que presenta al manejar grandes conjuntos de datos que no solo requieren de visualización, sino también de recursos y técnicas avanzadas como el sistema de visualización a distancia y análisis de datos desarrollado en TACC [12] o los utilizados en investigación [13] por las universidades para distribuir imágenes en pantallas con la herramienta HIPerWorks, también se encuentran los sistemas para imágenes, video, datos como ofrece la plataforma CSIROvision [14] que incorpora instalaciones de teleconferencia y aplicaciones interactivas permitiendo a los investigadores trabajar y manipular el mismo material en tiempo real.

En este proyecto se trata de ejecutar esta idea realizando interacciones con gráficos tridimensionales y texturas muy parecidas a las de un ser humano, su manipulación y posterior visualización en los display de un Clúster teniendo en cuenta que no solo

se tratan de imágenes 3D sino de una aplicación 3D. Ésta interacción requiere una programación de fácil entendimiento y con código accesible.

2.2 CONCEPTUAL

Simulación virtual en medicina. El área de la medicina ha conducido a la posibilidad de crear un mundo virtual que solamente existe en el computador de un cuerpo humano listo para ser explorado. Por lo que es posible su manipulación en cirugías no-intrusivas² minimizando de esta manera los trastornos corporales de las intervenciones y disminuyendo los riesgos, por ello es favorable simular en un sistema 3D.

Tridimensionales. Es una representación visual que contiene largo, ancho y profundidad de una imagen.

Textura. Hace referencia al aspecto o propiedad que tienen las superficies externas de los objetos.

Dispositivo electrónico. Es la combinación de varios componentes electrónicos destinados a controlar y aprovechar las señales eléctricas, haciendo referencia al teclado y mouse así como al proyecto de un guante electrónico elaborado por los ing. Robayo Sixto– ing. Pillajo Marcelo de la Universidad Politécnica Salesiana, Quito - Campus Sur.

Interacción. Permite al usuario manipular el curso de la acción dentro de una aplicación de realidad virtual, permitiendo que el sistema responda a los estímulos de la persona que la utiliza.

Render: Se refiere al proceso de generar una imagen desde un modelo y el tiempo de render depende de los parámetros establecidos en los materiales y luces, así como de la configuración del programa de renderizado.

² No-intrusiva: No hay disección de las cavidades del organismo como en una cirugía.

Clúster. Es un sistema de procesamiento de tipo paralelo o distribuido, que está formado de un conjunto de ordenadores conectados en red, que se comportan como uno solo con más capacidad de cómputo y con prestaciones avanzadas.

CGLX. Se encarga de la comunicación, corre la misma aplicación en el FrontEnd y en cada nodo en un clúster de visualización y utiliza la tarjeta gráfica local con toda la aceleración de hardware disponible.

Superusuario. Conocido también como root, nombre de la cuenta de usuario que posee todos los derechos de ejecución en un Sistema Operativo Linux.

Tiles. Conocido también como muros, wall o display y son los nodos en un clúster de visualización.

Wireless. Término usado para describir las telecomunicaciones en las cuales las ondas electromagnéticas (en vez de cables) llevan la señal, sobre parte o toda la trayectoria de la comunicación.

El desarrollo de la aplicación se lo hará en código abierto bajo el Sistema Operativo Linux, se utilizarán lenguajes de programación como C++, Java, HTML y librerías específicas para cubrir las necesidades del sistema a implementarse.

Para el desarrollo del aplicativo se tomará como guía una Metodología Ágil ya que se ajusta a las características del proyecto.

2.3 MARCO TEÓRICO

2.3.1 Metodología XP (Programación Extrema)

La metodología XP[8] se considera una de las más utilizadas, se la ha elegido debido a que se adapta a los requerimientos del proyecto “Sistema de Entrenamiento Virtual para Medicina”, algunas de sus características como: agilidad, ligereza y sencillez.

Características:[2]

- **Simplicidad:** Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento tanto en el código como en la documentación, eligiendo adecuadamente los nombres de las variables, métodos y clases, asegurándose que cuanto más grande se haga la aplicación, todo el equipo conocerá mejor el sistema.
- **Comunicación:** A los programadores les resulta de fácil manejo, código simple, documentado y comentando. Entre desarrolladores y cliente es fundamental el diálogo ya que de la buena comunicación entre las dos partes dependerá el éxito o fracaso de la aplicación en desarrollo.
- **Retroalimentación:** Se buscará la opinión del cliente para minimizar el trabajo al momento de realizar partes que no cumplan con los requisitos y evitar malentendidos en el desarrollo del aplicativo.
- **Valentía:** Para saber cuándo desechar un código obsoleto aunque haya significado esfuerzo y tiempo, también ser persistente ante problemas complejos que pueden hacer permanecer sin avanzar un tiempo pero luego serán resueltos ágilmente.
- **Respeto:** Los miembros del equipo buscan la calidad con un diseño óptimo y eficiente para el aplicativo respetando el trabajo de los demás y elevando el ritmo de producción en el equipo.
- **Desarrollo iterativo e incremental:** Se harán mejoras aunque pequeñas pero unas tras otras.
- **Pruebas unitarias continuas:** Se lo hará con frecuencia de forma manual y a veces automatizada.
- **Programación en parejas:** Mediante duplos de personas y permitiendo unir código realizado por separado y programando conjuntamente, permite llevar a cabo una mayor calidad del código.
- **Frecuente integración del equipo de programación con el cliente:** Para dudas, cambios o mejoras en el aplicativo 3D.

- **Corrección de todos los errores:** Dejándolo funcional antes de continuar con una nueva etapa en el proyecto.
- **Refactorización del código:** Reescribiendo ciertas partes del código para aumentar la funcionalidad sin modificar su comportamiento.
- **Propiedad del código compartida:** Los programadores podrán corregir y mejorar cualquier parte del proyecto puesto que tiene relativa facilidad en el entendimiento del mismo.

Fases:

A. Primera: Planificación Del Proyecto

- **Historias de usuario:** Similar a los casos de uso, especifica los requisitos del software. Se utiliza para apreciar el tiempo de desarrollo de la parte de aplicación, estimando la duración de cada una de ellas las cuales pueden ser cambiadas, mejoradas o eliminadas según lo requiera el cliente.
- **Plan de entrega:** Consiste en un plan de publicaciones en la cual se especifican las historias que se establecieron para cada versión del programa.
- **Iteraciones:** Según las historias de usuarios se implementa el código y se lo reutiliza para mejorar el aplicativo.
- **Velocidad del proyecto:** Representa la velocidad a la que se desarrolla el proyecto estimada en base al número de historias que se implementaron en las iteraciones.
- **Programación en pareja:** Incrementa la productividad y calidad del aplicativo, puesto que simultáneamente es desarrollado, revisado y discutido disminuyendo drásticamente errores y sacando un producto efectivo.
- **Reuniones diarias:** Son necesarias para la exposición de los problemas y así poder encontrar ideas que ayuden a una posible solución.

B. Segunda: Diseño

- **Diseños simples:** Esta metodología sugiere simplicidad en los diseños para alcanzar una implementación rápida y entendible.

- **Glosario de términos:** Teniendo una descripción adecuada de los nombres de los métodos y clases ayudarán a comprender el diseño, facilitando la reutilización de código para posteriores mejoras en el aplicativo.
- **Riesgos:** En el desarrollo del proyecto pueden surgir problemas por lo que la XP recomienda que una pareja investigue disminuyendo así el riesgo que pueden producir estos problemas.
- **Funcionalidad extra:** Realizar solo lo que el cliente ha solicitado ya que las funciones extras restan tiempo y recursos.
- **Refactorizar:** Revisar el código generado y mejorarlo para optimizar su funcionamiento pero sin cambiar su funcionalidad.
- **Tarjetas C.R.C. (clase, responsabilidad, colaboración):** Destaca la programación orientada a objetos.

C. Tercera: Codificación

XP propone que los equipos de desarrolladores publiquen cada cierto período de tiempo sus códigos implementados y corregidos para que de esta manera todos cuenten con códigos que les puedan ser útiles para resolver problemas.

D. Cuarta: Pruebas

- **Pruebas de unidad:** Serán implementadas en un marco de trabajo en el que sean automatizadas para ser ejecutadas de manera fácil y repetida.
- **Pruebas de aceptación:** Se enfoca en las características generales y en la funcionalidad del sistema ya que se derivan de las historias de usuarios que se implementaron.

2.3.2 Entorno para desarrollo 3D

Lenguajes informáticos

Están comprendidos los lenguajes de programación y de marcado que se los utilizará para la elaboración del aplicativo 3D y su inclusión en la Web, así como también para el procesamiento paralelo.

A. Lenguajes de marcado[15]:

Es un conjunto de instrucciones que permiten diseñar el contenido de los documentos incorporando etiquetas o marcas los cuales contendrán información de su presentación, por lo que será usado para la exposición del aplicativo 3D interactuando junto al dispositivo electrónico a través de la Web.

En la evolución de la Web los CGI (Common Gateway Interface) definen el mecanismo por el cual se puede pasar información entre el servidor y ciertos programas externos, dando un carácter dinámico a las páginas, pero el inconveniente es su rendimiento ya que al momento de recibir varias peticiones al mismo tiempo no puede soportar la carga.

El aumento en el número de arquitecturas y lenguajes de programación permiten el desarrollo de aplicaciones Web, para soportar la carga de trabajo generada por la aplicación. La solución está en que el diseño del sistema de ejecución quede mejor integrado con el servidor, permitiendo integrar aplicativos que interpreten comandos incrustados en las páginas HTML y dotando a los servidores la ejecución de programas mejor enlazados, de forma que el servidor lo ejecute optimizando el rendimiento.

Cuadro Comparativo

Lenguajes de Marcado			
Características	HTML	BBCode	XML
Software Libre	X	X	X
Portable entre distintas plataformas	X	X	X
Soporta múltiples servidores Web	X	X	X
Conectividad con varias Bases de Datos	X		X
Permisibilidad en la construcción del documento	X		
Curva de aprendizaje	X	X	X
Mayor información para incorporar aplicativos en su entorno	X		
No se pueden combinar elementos en			X

Lenguajes de Marcado			
Características	HTML	BBCode	XML
diferentes vocabularios			
Proporcionar al usuario un objeto de tipo Gráfico sobre el cual dibujar	X		
Permite incrustar un Applet	X		

Tabla 2-1 Cuadro comparativo entre los diferentes lenguajes de marcado

Fuente: Autores

1) Lenguaje HTML

Sus siglas en inglés *HypertextMarkupLanguage*, es el lenguaje utilizado para el diseño de las Páginas Web o de Internet. En español su traducción vendría a ser lenguaje de etiquetado de documentos hipertextual[16].

Características:[17]

- Es sencillo y se lo puede crear en cualquier editor de texto común.
- Es un estándar cuyas normas lo define un organismo llamado World Wide Web Consortium (W3C), una página HTML se visualiza de forma muy similar en cualquier navegador de cualquier Sistema Operativo, ya que es un lenguaje reconocido universalmente y que permite publicar información de forma global
- Es un lenguaje utilizado únicamente para dar estructura a una página Web, el estilo de la propia página vendrá dado por un enlace a una hoja CSS (hojas de estilo en cascada). Es decir, que toda etiqueta que defina estilo y no estructura no se podrá considerar perteneciente al lenguaje HTML.
- Utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determina la forma en la que debe aparecer en su navegador el texto, así como también las imágenes y los demás elementos, en la pantalla del ordenador.
- Cada elemento de un documento HTML consta de una etiqueta de comienzo, un bloque de texto y una etiqueta de fin
- Las etiquetas tienen que seguir un orden piramidal, las primeras que se abren son las últimas que se cierran.

La función principal de esta aplicación basada en el lenguaje HTML, es proporcionar al usuario un objeto de tipo gráfico sobre el cual dibujar y varias funciones para facilitar el uso del mismo.

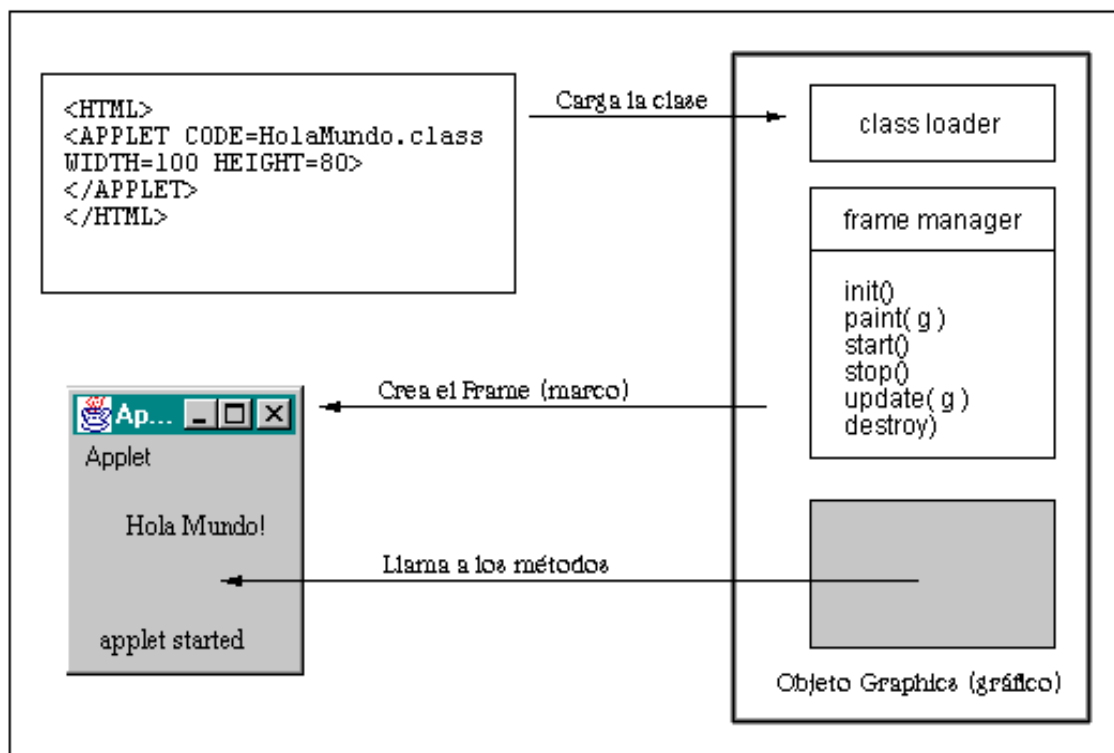


Figura 2-1 Arquitectura HTML – Applet

Fuente: Froufe Agustín, Tutorial de Java - Arquitectura de appletviewer,2007[18]

2) Lenguaje BBCode

BBCode[19] es un lenguaje de marcado utilizado en foros y e-mails para dar una mejor apariencia a un mensaje o post presentado. Se encuentra basado en el lenguaje HTML muy similar en estructura y sintaxis, siendo una versión simplificada por lo que resulta sencillo de aprender y utilizar.

La estructura básica del lenguaje tiene propiedades de contenido y atributo:

- Los elementos poseen una etiqueta de inicio y una de cierre donde el contenido queda enmarcado entre ambas.
- Los atributos son valores escritos en la etiqueta de comienzo de un elemento y asignados a la misma por un signo de igualdad después del nombre de esta.

BBCode debe ser procesado por un script de análisis o *parser* del lado del servidor donde se encuentra el programa para que este, analice el texto y lo entregue al navegador en un formato adecuado, es decir, no solo texto.

Características:

- Función de fragmentos de código: permite insertar rápidamente frases favoritas presionando las teclas Ctrl + espacio.
- Vista previa en vivo: Los textos pueden ser visualizados de forma automática en el navegador.
- Corrector ortográfico: Ayuda en la corrección de errores de ortografía.
- Asistente de Tags: Para insertar fácilmente imágenes, listas, tablas.
- Botones y opciones de BBCode personalizados: Los usuarios pueden añadir sus propios botones en la barra de herramientas.
- Posee modificadores de apariencia de texto
- Inserta hipervínculos con facilidad utilizando la etiqueta URL.
- Posee estructuras ordenadas como listas y tablas similares a HTML
- Presentación de imágenes a través del marcador img.

3) Lenguaje XML

XML (eXtensibleMarkupLanguage): Es un meta-lenguaje que permite crear etiquetas adaptadas a las necesidades y fue desarrollado por el World Wide Web Consortium. Es estricto en cuanto a lo que está permitido y lo que no, cumpliendo todo documento con dos condiciones: ser válido y estar bien formado.

Todo documento XML es a su vez SGML, donde los programas y documentos creados con SGML pueden convertirse al nuevo lenguaje simplificando la complejidad, facilitando el aprendizaje y la implementación del nuevo estándar. Este lenguaje logra un equilibrio entre simplicidad y flexibilidad.

Es un lenguaje desarrollado para permitir la descripción de información contenida en el WWW a través de estándares y formatos comunes, de manera que tanto los

usuarios de Internet como programas específicos (agentes) puedan buscar, comparar y compartir información en la red.

Al utilizar los formatos de la industria XML y XML-based para el intercambio de información normalizada, permiten la creación de aplicaciones que utilizan la arquitectura XML end-to-end para almacenar, intercambiar, ver y manipular XML.

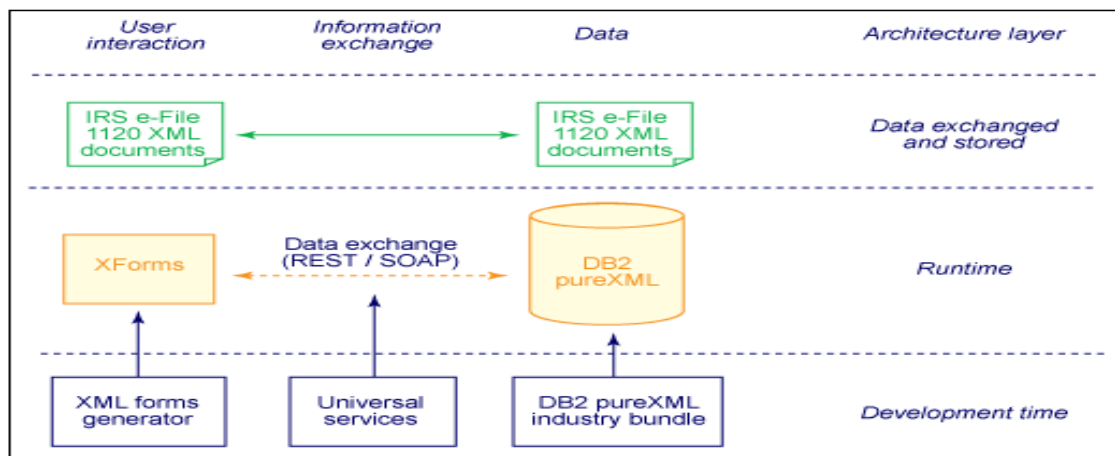


Figura 2-2 Arquitectura XML

Fuente: IBM, Build a DB2 pureXML application in a day, 2008[20]

Características:

- Permite proporcionar diferentes vistas sobre los datos (HTML, PDF, voz), dependiendo de quién sea el cliente.
- Facilita la integración desde fuentes de datos heterogéneas, ejemplo: páginas Web, distintas bases de datos.
- XML es un estándar para escribir datos estructurados en un fichero de texto proporcionando un conjunto de reglas, normas y convenciones para diseñar formatos de texto que van desde las hojas de cálculo hasta parámetros de configuración, transacciones financieras o dibujos técnicos.
- Parece HTML por usar marcas y atributos pero no lo es, en XML sólo se usan marcas para delimitar fragmentos de datos, dejando la interpretación de éstos a la aplicación que los lee.

- XML está en formato texto para ser usado en cualquier plataforma, proporcionando innumerables ventajas de portabilidad, depuración, independencia de plataforma, e incluso de edición, pero no hay permisividad en la construcción de documentos.
- No requiere licencia siendo un estándar abierto independiente de la plataforma, con soporte para varias herramientas.
- Es un subconjunto de SGML siendo una tecnología madura.
- Soporta *unicode*.
- Se orienta a los datos y su semántica, no a la representación.
- Se está convirtiendo en el lenguaje de bases de datos de la Web.
- Permite un fácil intercambio de información entre aplicaciones.
- Al tratarse de un metalenguaje tiene un vocabulario extensible.
- Permite definir lenguajes de marcado por medio de DTD's (DocumentTypeDefinition) o de XML-Schemas

El XML fue ideado en principio para entornos semi-estructurados, como textos y publicaciones. Los lenguajes basados en XML tienen algunas aplicaciones como por ejemplo en la transacción de datos entre servidores, intercambio de información financiera, fórmulas, reacciones químicas.

B. Lenguajes de programación[21]:

Es un lenguaje diseñado para controlar el comportamiento de una máquina (computadora), es decir, es un modo práctico para poder dar instrucciones a un equipo y que este los ejecute.

Para ser elaborado el aplicativo 3D a continuación se muestra las características de algunos lenguajes de programación de los cuales se seleccionará posteriormente el que mejor se adapte a las necesidades del aplicativo 3D a desarrollar:

Lenguajes de Programación			
Características	Python	Java	C++
Fácil de aprender.		X	X
Versión del sistema poco intuitivo	X		
Multiplataforma		X	X
Se orienta a objetos	X	X	X
Es sencillo, robusto, seguro, portable, Neutral, hilo (Threads).		X	
Se orienta a la implementación de sistemas operativos. Se utiliza para crear software de sistemas y aplicaciones.	X		X
Tiene algunas excepciones y una representación alta.		X	X
Permite interactuar y enlazar varios sistemas	X	X	
Muy bueno para proyectos grandes o aquellos en donde se utiliza mucho código		X	
No existen referencias		X	
Programación Genérica ³			X
Lenguajes de uso más extendido, por lo que resulta fácil encontrar información, documentación y fuentes para los proyectos		X	X
Mal manejo de archivo de datos	X		
Solución de múltiples dependencias en el sistema operativo para lanzar una aplicación	X		

Tabla 2-2 Cuadro Comparativo – Características de Lenguajes de Programación

Fuente: Autores

1) Lenguaje Python

Es un lenguaje de programación orientado a objetos, admite un embalaje jerárquico de código modular por lo que es popular en el desarrollo de software científico. Desde el 2005 los desarrolladores han codificado envoltorios para varios paquetes 3D permitiendo servir estos paquetes como anfitriones ePMV.

Estas envolturas exponen una interfaz de un programa de aplicación (API) para el intérprete de Python[22], por tanto una incorporación de un intérprete puede integrar

³Programación genérica: Generaliza las funciones utilizadas para que puedan usarse en más de una ocasión.

todo el lenguaje de programación, además de las aplicaciones codificadas dentro de su propia API.

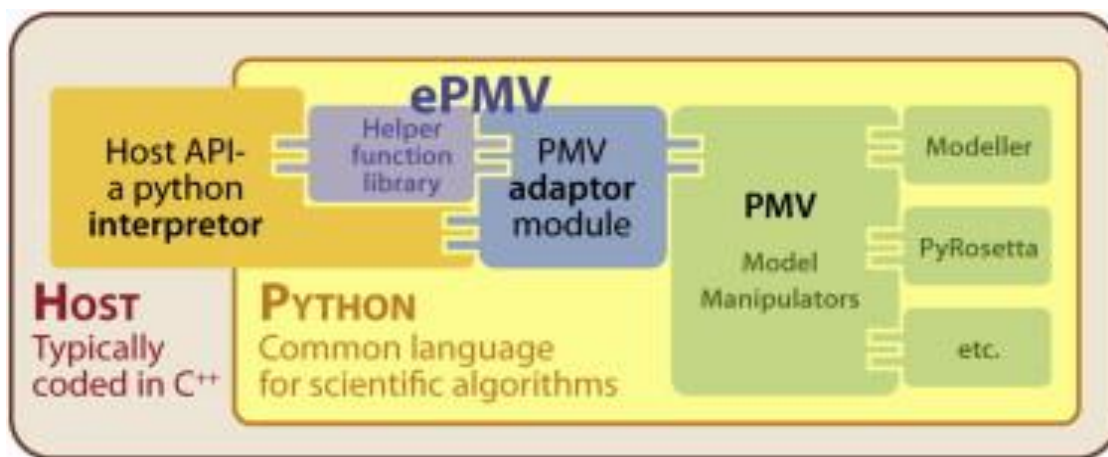


Figura 2-3 Arquitectura de aplicación host que incrusta aplicaciones Python

Fuente: The Scripps Research Institute, ePMV Embeds Molecular Modeling into Professional Animation Software Environments, 2011[23]

Con una base común de Python ePMV puede traducir la lógica, la arquitectura y las variaciones de mando entre el host y el PMV.

2) Lenguaje C++

Uno de los objetivos de C++[24] es que son necesarias pocas instrucciones en el lenguaje de máquina para traducir los elementos del lenguaje, sin que sea necesario un soporte intenso en tiempo de ejecución, fue creado junto con el sistema operativo Unix, para ser usado por programadores.

Una de las ventajas es su rica colección de clases y funciones que existen dentro de la biblioteca estándar de C++ y al crear estas propias se sabrá exactamente cómo funcionan y se podrá examinar el código de C++. Las extensiones comunes de este lenguaje son: .h .hh .hpp .hxx .h++ .cc .cpp .cxx .c++.

El procesador es un programa de utilidad que procesa instrucciones y analiza cada declaración como el tipo, variables y declaraciones de funciones, incluidas las

definiciones *metanivel*⁴. Estas instrucciones pueden estar incluyendo una librería o algunas instrucciones especiales al compilar algunos términos utilizados en el programa.

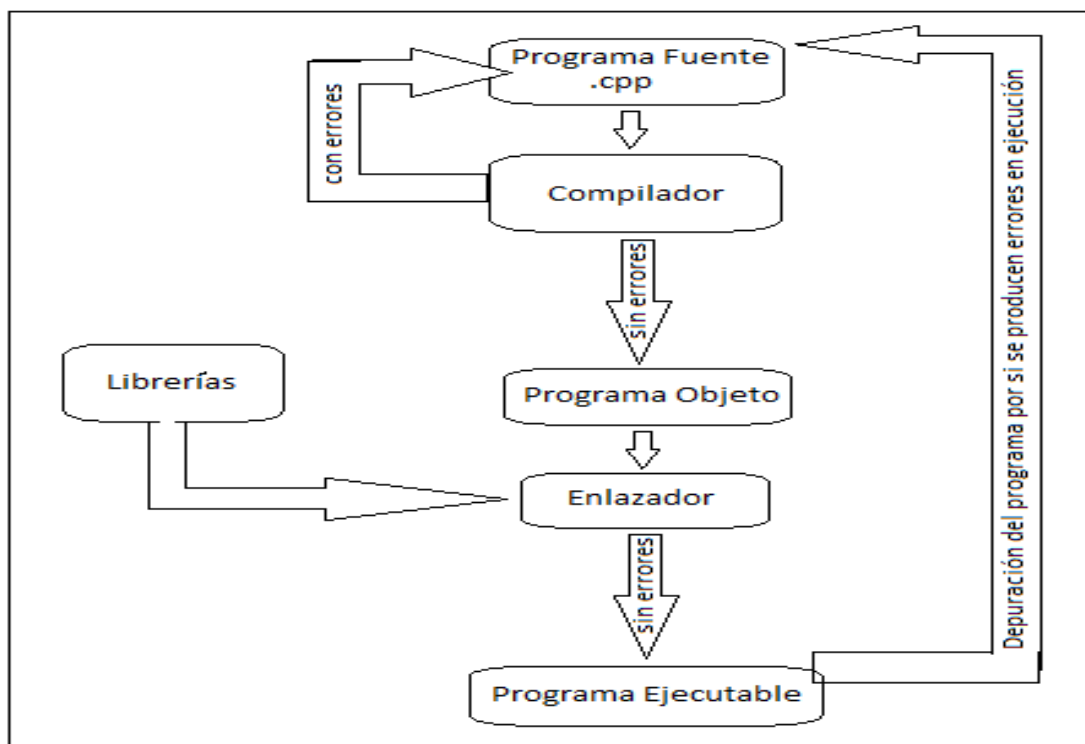


Figura 2-4 Arquitectura C++

Fuente: Coding Unit, C++ Preprocessor Directives, 2012[25]

3) Lenguaje Java

Java[26] es un lenguaje simple orientado a objetos, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, multihilos y dinámico.

Para establecer Java como parte integral de la red, el código fuente Java se compila a un código de bytes de alto nivel independiente de la arquitectura de la máquina en la que se ejecutará. Este código (*ByteCode*⁵) está diseñado para ejecutarse en una

⁴Metanivel:proporciona información sobre propiedades seleccionadas del sistema y hace el software auto-consciente

⁵Bytecode: Es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un archivo binario que contiene un programa ejecutable similar a un módulo objeto, que es un archivo binario producido por el compilador cuyo contenido es el código objeto o código máquina

máquina hipotética que es implementada por un sistema run-time, el cual si es dependiente de la plataforma.

La máquina virtual Java y las librerías son dependientes del sistema y permitirían acceder directamente al hardware de la máquina.

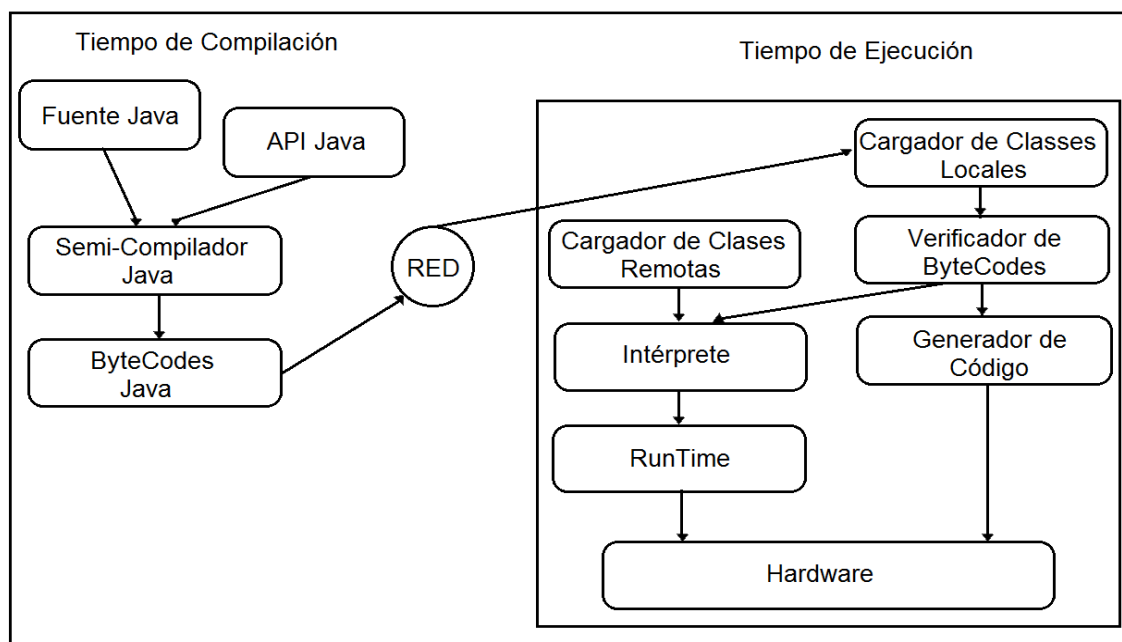


Figura 2-5 Arquitectura Java

Fuente: Froufe Agustín, Tutorial de Java - Características de Java, 1999[27]

Las extensiones comunes dentro del lenguaje son: .java, .class, .jar.

Java aporta dentro de los navegadores la capacidad de desplazar el control de la interactividad de los servidores hacia las máquinas de los usuarios que se utilizan para recorrer Internet.

La estructura del lenguaje, permite a los navegadores la tele-carga⁶ de Applets, estos fragmentos de programa son compactos y pre-compilados, que pueden interpretar de modo distinto los datos tele-cargados para producir por ejemplo animaciones, sonido y especialmente la verdadera interactividad. El lenguaje Java, contemplado desde un

⁶Tele-carga: Es una forma de decir que han instalado un nuevo software en la central

navegador de Internet, pues no es ni totalmente interpretado⁷, ni totalmente compilado⁸, se transforma en un código elemental parecido al ensamblador, llamado también *p-code* o *byte-code*.

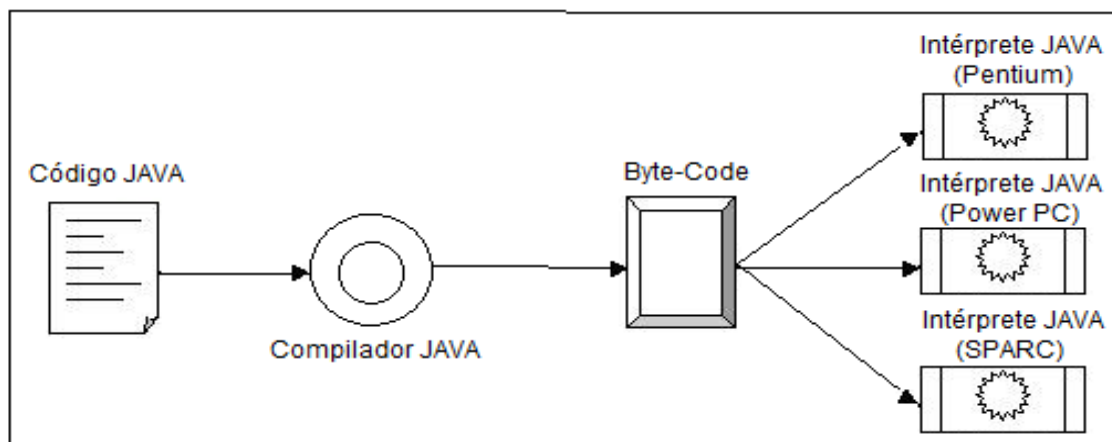


Figura 2-6 Transformación para la máquina virtual

Fuente: Centro de tecnología informática (SAO), Introducción al lenguaje JAVA,2000[28].

Tiene la particularidad de ser compacto y por tanto puede ser compilado (traducido a lenguaje máquina) muy rápidamente, en el transcurso de la propia ejecución del programa.

Los gráficos 3D tienen varios campos como: la ingeniería o la ciencia y se desarrollan y usan originalmente en estas áreas, utilizando carísimas estaciones de trabajo, por ello se utiliza Open GL puesto que esta API es robusta y óptima para los ambientes virtuales 3D.

C. Bibliotecas para entorno 3D:

OpenGL: Es un estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Esta librería gráfica permite crear dibujos 3D en varios sistemas y entornos de desarrollo, provee al programador una serie de primitivas que le permiten definir un entorno tridimensional, dibujar objetos geométricos en su interior, agregarles luz y color, mejorar la calidad visual

⁷ Interpretado: Se compila cuando se corre por medio de un programa que reside en la máquina donde se ejecuta dicho código fuente, traduciéndolo y estableciendo la línea que corresponda durante la ejecución.

⁸ Compilado: Pasó por un proceso que termina en un archivo de código binario creando un ejecutable

mediante algunas técnicas avanzadas y asignar texturas a los objetos para obtener efectos más realistas.

Características:

- Es una biblioteca desarrollada por SGI[29], la mayor organización especializada en software gráfico en el mundo.
- Es portable y rápido: existen APIs de OpenGL para C++, Java, Visual Basic y otros, el código es portable incluso a distintos sistemas operativos como Windows, Linux/Unix, MacOS.
- Es procedural y no descriptivo, es decir, el programador no describe una escena sino los objetos de la misma y los pasos necesarios para configurarla.
- Actúa en modo inmediato, es decir, los objetos son dibujados conforme van creándose.
- Es el estándar actual en la industria de los gráficos 3D.
- Maneja los gráficos a muy bajo nivel proporcionando gran libertad de acción al programador, siendo fácil de usar.

OpenGL envía directrices al Sistema Operativo para indicarle que el enlace entre el programador y este se realiza mediante una interfaz eficiente y portable.

Esquemmatizando su funcionamiento:

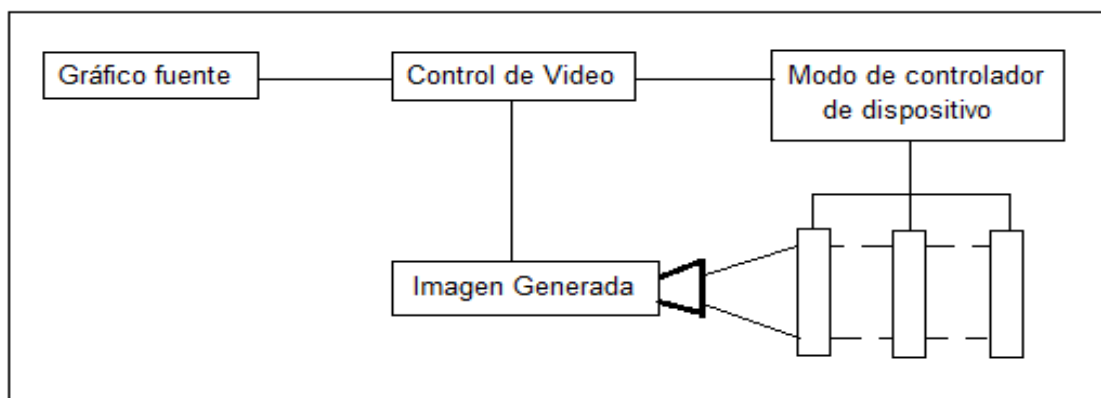


Figura 2-7 Esquema OpenGL

Fuente: Snuffer John T., System and method of extracting 3-D data generated for 2-D display applications for use in 3-D volumetric displays, 2004[30]

Mesa: Se trata de una librería cuyo API es básicamente igual que el de OpenGL incorporando toda la funcionalidad de esta, siendo una implementación de fuente abierta/libre.

Es compatible con la representación de software puro y proporciona aceleración por hardware para varias tarjetas gráficas 3D bajo Linux.

Incorpora toda la funcionalidad de OpenGL con la única excepción de algunas rutinas muy específicas, que se refieren a:

- *NURBS* recortadas
- *Antialiasing* para polígonos

No existen diferencias sustanciales entre mesa y OpenGL

Estructura básica de OpenGL

- OpenGL es un API, no un lenguaje de programación por lo que se requiere de un lenguaje como C o C++.
- Es independiente del sistema de ventanas utilizado (no incorpora rutinas para su manejo) y del sistema Operativo.
- Incorpora la funcionalidad básica para realizar la visualización (*rendering*), no obstante existe un conjunto de aspectos que no son estrictamente de visualización y que se encuentran en una librería estándar adicional llamada GLU (*OpenGL Utility Library*).
- Existen algunas herramientas que facilitan el trabajo de adherir OpenGL con un entorno de ventanas. Un ejemplo es GLUT (*OpenGL Utility Toolkit*), un completo conjunto de herramientas que facilitan la labor con OpenGL y MS-Windows o XWindows.

Los desarrolladores utilizan OpenGL con sus bibliotecas de representación robusta para un mayor nivel de APIs, algunas de las cuales son:

Bibliotecas					
Características	Open Inventor	VTK	Open Scene Graph	Java 3D	CGL X
Aplicado en el campo de Visualización científica	X	X	X	X	X
Aplicado en el campo de videojuegos			X	X	X
Aplicado en la educación		X	X	X	X
Aplicado en el campo de Visualización médica	X	X	X	X	X
Programación orientada a objetos 3D	X	X	X	X	X
La comunidad de desarrolladores es muy activa y se puede conseguir ayuda en las listas de correos			X	X	
Código portable	X	X	X		X
El aplicativo se integra con facilidad en una página Web				X	
Software libre	X	X	X	X	X
Procesamiento de imágenes 2D/3D	X	X	X	X	X

Tabla 2-3 Bibliotecas OpenGL y plataformas soportadas

Fuente: Autores

1) Open Inventor

Consiste en un conjunto de clases en lenguaje C++, .NET y Java, que permiten diseñar e implementar aplicaciones que sigan el paradigma de la programación orientada a objetos. Con Open Inventor[31] es necesario código adicional para animar partes de la escena, la capacidad de hacer estos cambios está integrado en el modelo de programación.

Open Inventor es un conjunto de bloques de construcción que le permite escribir programas que aprovechen las poderosas características de hardware de gráficos con el esfuerzo de programación mínima. Sobre la base de OpenGL, el conjunto de herramientas proporciona una biblioteca de objetos que se pueden utilizar, modificar

y ampliar para satisfacer las nuevas necesidades. Inventor incluye objetos primitivos bases de datos, incluida la forma, posición, grupo, objetos y máquinas; manipuladores interactivos, como la barra de herramientas flotante y el trackball, y componentes, como el editor de materiales, editor luz direccional, y el espectador examinador.

La arquitectura de una aplicación Open Inventor se muestra en la Figura 2-8:

- El kit de herramientas de Open Inventor proporciona tres herramientas de programación que se pueden utilizar en la aplicación.
- La librería de componentes es una biblioteca de comodidad para los programadores que utilizan el sistema de ventanas X y basados en herramientas tales como *Xt* y *Motif*.
- Incluye un formato de intercambio de archivos para el cambio de objetos y escenas 3D entre aplicaciones.
- El IRIS *Graphics Library* (IRIS GL) es una biblioteca de subrutinas para la creación de gráficos a color en 2D, 3D y animación.

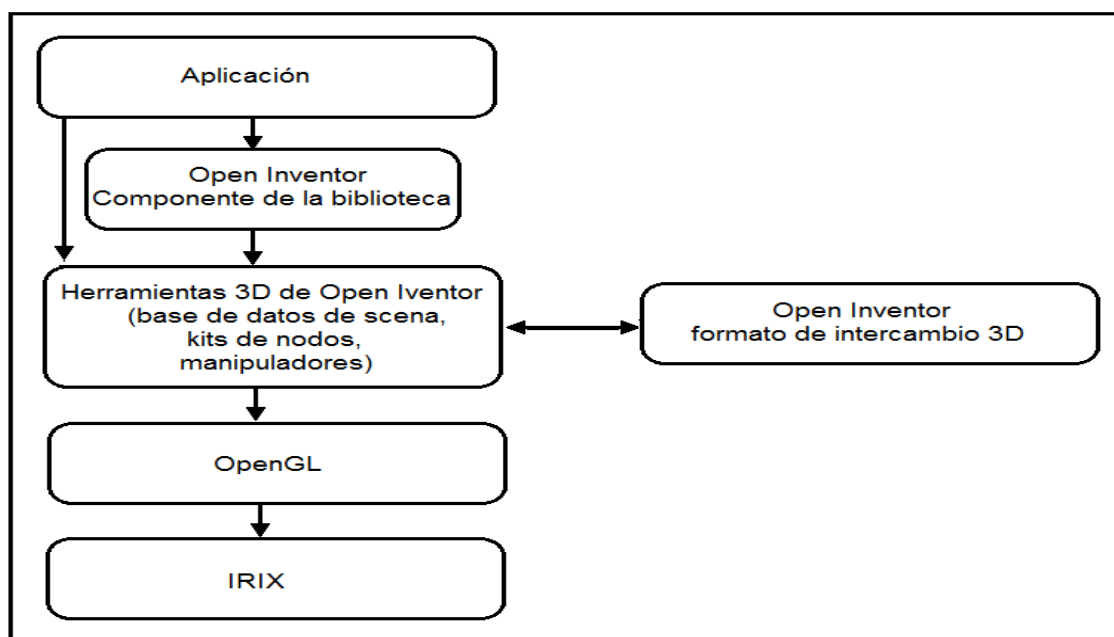


Figura 2-8 Arquitectura Open Inventor

Fuente: Silicon Graphics, Chapter 6. Application Libraries, 1998[32]

2) VTK

El kit de herramientas de visualización (VTK)[33] es código abierto y biblioteca de software libre bajo la licencia BSD de clase C ++ para la visualización de datos y procesamiento de imágenes 2D/3D por computadora. VTK consiste en una librería de C ++ compatible con una amplia variedad de algoritmos de visualización. VTK es multiplataforma y soporta el procesamiento paralelo, además se integra con diversas bases de datos en herramientas GUI como Qt y Tk, funciona en Linux, Windows, Mac y Unix.[34]

Características:

- Capacidades de Graficación y Render ya que incluye capacidades para desplegar primitivas geométricas, también incluye algoritmos para la visualización de campos escalares, campos vectoriales, visualización de tensores, mapeo de texturas y métodos volumétricos, además incluye técnicas avanzadas de modelado tales como: modelado implícito, la reducción de polígonos, suavizado de mallas, corte, el contorno y la triangulación de Delauny.
- Capacidad de desarrollo de la interfaz de usuario con algunos lenguajes como: Tcl/Tk, Python/Tk y Java. De código Libre (Open Source), independiente de la plataforma o biblioteca.
- Extensible, portable, basado en el estándar OpenGL porque brinda un alto nivel de abstracción para gráficos 3D.
- Integrado con una biblioteca de interfaces gráficas de usuario (Tk, Qt, FLTK, WxWidgets, Java, X11, Windows, Mac OS).
- Posee varios formatos de importación / exportación para imágenes 2D/3D, también una versión paralela y multiprocesos.

La arquitectura de VTK:

- La estructura central es una canalización de datos a partir de una fuente de información para una imagen mostrada en pantalla,

- Los módulos individuales no actualizan su salida hasta que son llamados a hacerlo por algún otro módulo.
- Posee un núcleo compuesto de clases C++ compilado y una capa interpretado Tcl-Tk, Python y Java

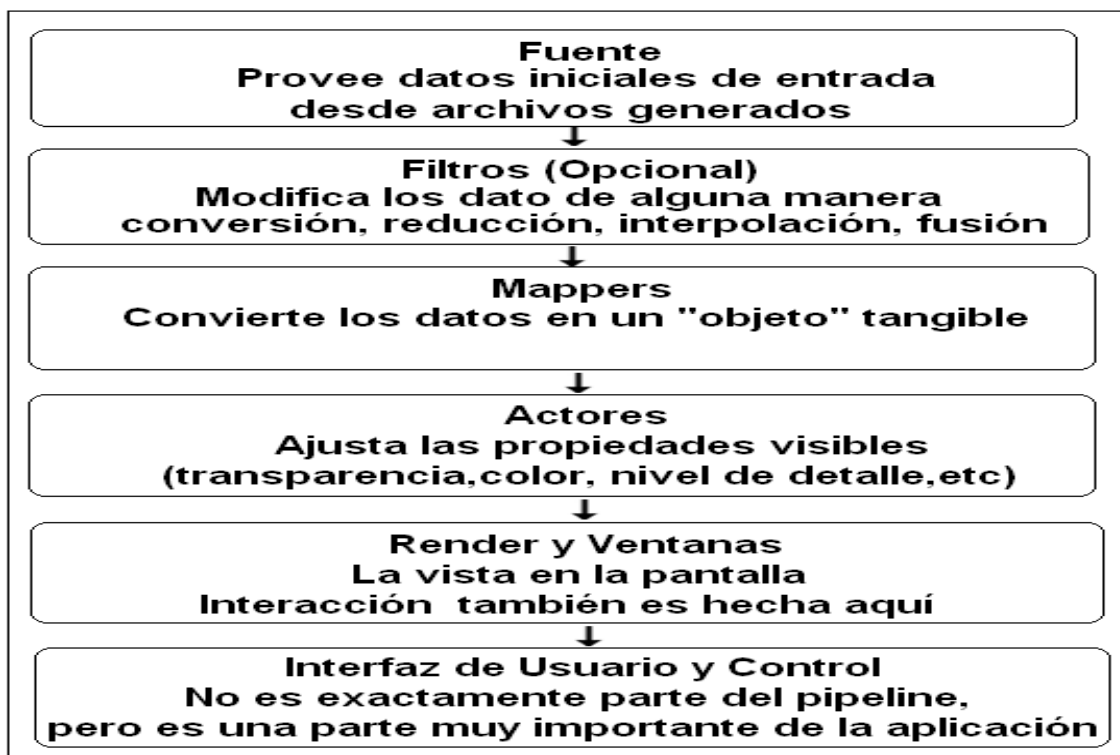


Figura 2-9 Arquitectura de Visualización Pipeline (Oleoducto) VTK

Fuente: Bell, VTK Tutorial, 2004[35]

3) OpenSceneGraph (OSG)

Está orientado a objetos y construido a partir de la librería OpenGL y C++ estándar el cual ofrece instalaciones para la prestación de los objetos en un espacio 3D. OSG[36] es un “*toolkit*” gráfico de código abierto que puede ser utilizado para representar una escena como una estructura de datos en forma de árbol y los nodos son los objetos de la escena, para el desarrollo de aplicaciones gráficas. Por lo general, una escena incluye una cámara, la iluminación, los modelos y texturas.

La arquitectura de OSG [37] está compuesta por el núcleo de OSG (core OSG), los *NodeKits* y los plugins, además de aplicaciones y ejemplos donde:

- El núcleo de OSG es donde están las librerías más útiles como: *osg*, *osgDB* que permite el manejo de plugins, *osgUtil*, *osgViewer* que maneja la escena, *osgGA* que permite el manejo de eventos, Open Threads provee la interface para programar en C++
- Los *NodeKits* son librerías extendidas del núcleo de OSG como: *osgAnimation*, *osgFX*, *osgGA*, *osgManipulator*, *osgParticle*, *osgShadow*, *osgSim*, *osgTerrain*, *osgText*, *osgViewer*, *osgVolume*, *osgWidget*, *osgQt*, que permite el manejo de efectos, manipular objetos, terrenos, sombras.
- Los plugins son para el soporte de entrada/salida en una aplicación para imágenes 2D o formatos 3D.

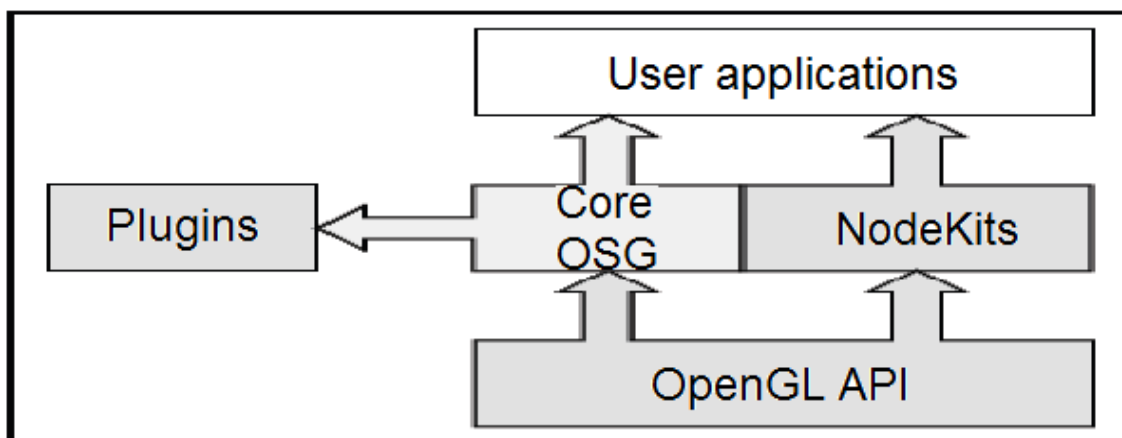


Figura2-10 Arquitectura OSG

Fuente: WANG Rui, QIAN Xuelei, OpenSceneGraph 3.0: Beginner's Guide, 2010

4) Java 3D

El API de Java 3D[38] tiene un conjunto de clases, interfaces y librerías de alto nivel las cuales permiten aprovechar la aceleración gráfica por hardware que incorporan las tarjetas gráficas y las llamadas a los métodos de Java 3D son transformadas en llamadas a funciones de OpenGL, esta herramienta proporciona la facilidad necesaria para la creación de mundos virtuales en tercera dimensión, ejecutándose como Applets alojados en navegadores de Internet o como aplicaciones solitarias.

La realidad virtual es un gran consumidor de cálculos. Java 3D utiliza las optimizaciones posibles 3D que ofrece el sistema en el que se ejecuta la máquina virtual de Java:

Cuando se utiliza una versión de Java 3D OpenGL, algunas características de los métodos de la API involucran al nativo de *Java DLL 3D/OpenGL*.

El sistema *DLL OpenGL* utiliza el acelerador de gráficos disponible en la tarjeta gráfica. El poder de procesamiento de Java 3D depende de la potencia de su procesador y de la tarjeta gráfica. Comprobar que este hardware tenga soporte o compatibilidad con los requerimientos de la aplicación y el sistema operativo para aprovechar al máximo su capacidad de procesamiento.

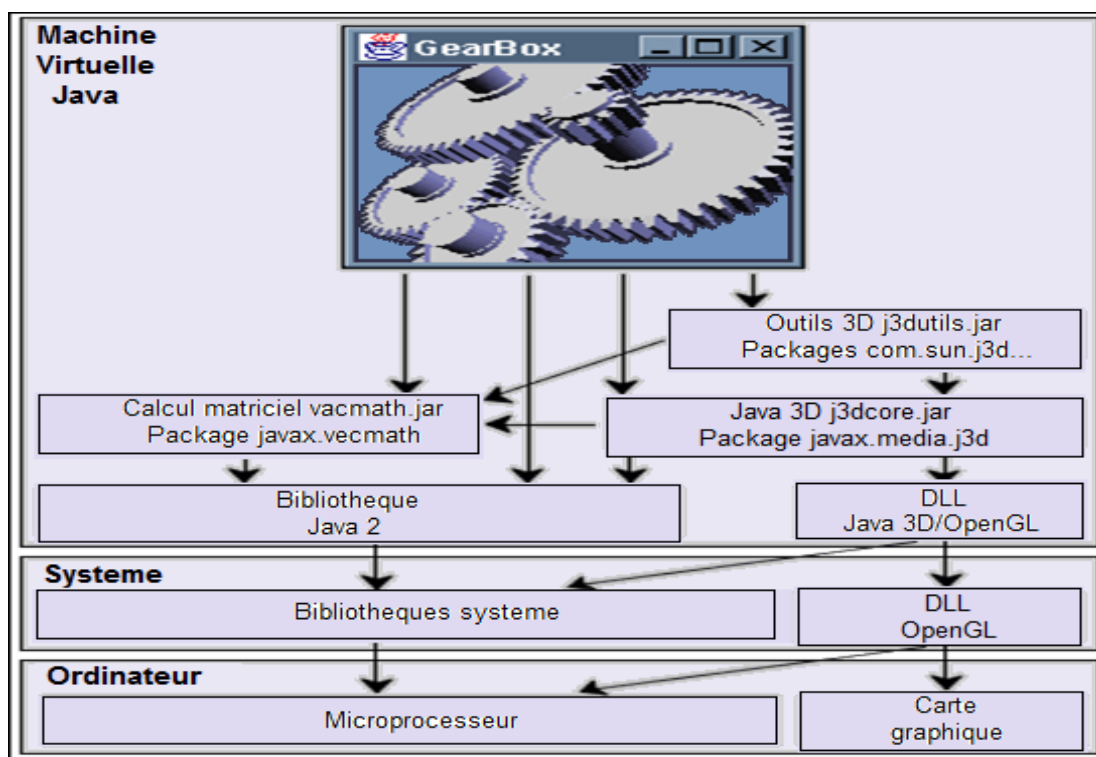


Figura 2-11 Arquitectura Java 3D

Fuente: PUYBARET, Java 3D, 2007[39]

Java 3D se da a conocer como una extensión del JDK. Para el Applet, es necesaria la instalación de los plugins adecuados en los navegadores de Internet y la instalación de Java 3D *runtime* que se encuentra disponible en la página de Sun[40].

5) CGLX

Fue desarrollado para superar algunos problemas relacionados con la visualización de rendimiento especialmente en clúster basado en aplicaciones OpenGL. La idea es que cada nodo en un clúster de visualización va a correr la misma aplicación y la utilización de la tarjeta gráfica local con toda la aceleración de hardware disponible. CGLX interceptará sólo una vista seleccionada llamadas a OpenGL, por volver a implementar. Internamente CGLX se encargará de toda la comunicación entre la aplicación en los nodos y el cabezal de aplicación. Toda la información que necesita para ser distribuidos entre los nodos serán manejados por CGLX.

CGLX permite que las aplicaciones OpenGL se muestren en clústeres de visualización, incluyendo *tiled* de visualización o sistemas multi-proyector, pero en un entorno de clúster no es obligatorio. En cuanto CGLX es conectado una infraestructura de software de administración especial, descrita como clúster de middleware implementa una biblioteca compartida *cglXlib*, que es requerida para todas las herramientas CGLX y aplicaciones, esta librería proporciona a los programadores control sobre los recursos de la red y la interacción persona – ordenador. CGLX soporta sistemas XWindows en plataformas UNIX y Mac OS X de forma nativa. En los sistemas operativos basados en X, CGLX utiliza GLX para la representación directa. En los sistemas MAC OS X la estructura envuelve el nativo AGL y una estructura carbono por lo que el API presentado tiene un aspecto común en todos los sistemas operativos y los cambios de código no son necesarios cuando se pasa de un sistema a otro, además la librería toma ventaja de las características implementadas en los controladores de la tarjeta gráfica.

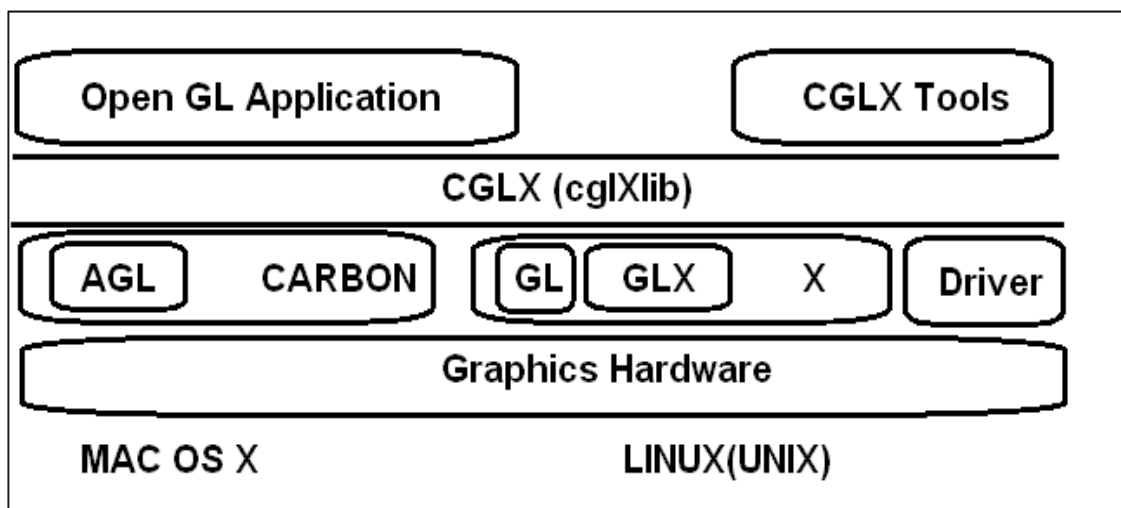


Figura 2-12 Arquitectura CGLX

Fuente: HlPerWorks, CGLX Documentation, 2012[41]

Para enlazar OpenGL y el Sistema Operativo es necesaria la librería GLX. Cada tarjeta gráfica 3D tiene su propia versión renovada para obtener un mejor rendimiento de los comandos OpenGL para dicha tarjeta. Por lo tanto, el programador simplemente escribe los comandos OpenGL y puede confiar en que su versión de “GLX.so” se ocupará de ejecutarlos en la forma más eficiente de acuerdo a la tarjeta gráfica que se encuentre instalada, a continuación se describe la librería más importante:

X11 (XWindow): Más conocido como XWindow o tan solo como X, es el servidor gráfico de Unix orientado a red y está basado en ventanas, en Ubuntu 11.04 por ejemplo usa la versión 11 de ahí que este paquete tiene el nombre de X11.

La extensión del Servidor X recibe los comandos de reproducción en el cliente y pasa a la biblioteca OpenGL instalado.

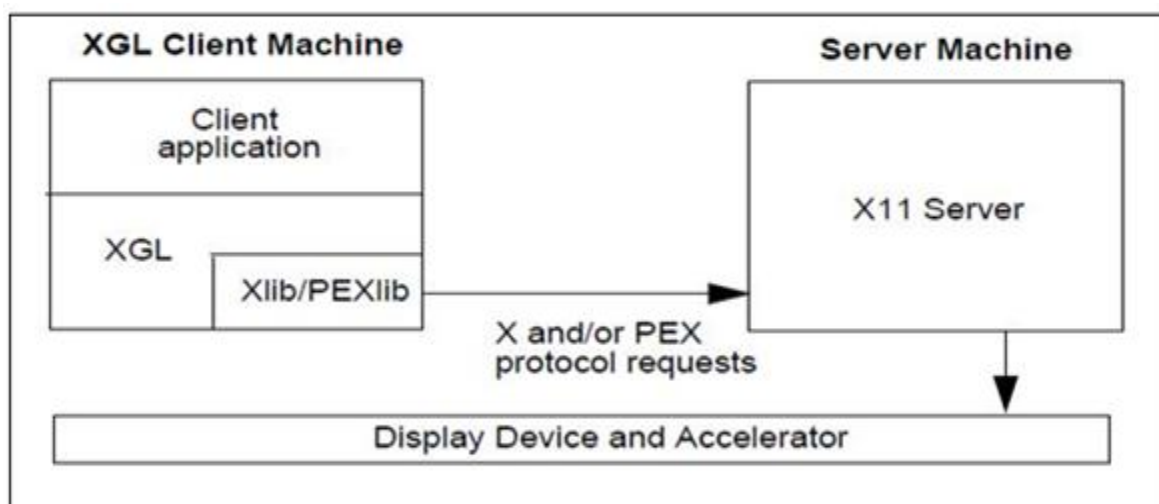


Figura 2-13 Arquitectura X11

Fuente: OpenGL, OpenGL – Kcchao, 2010[42]

D. Entorno de desarrollo integrado (IDE):

Es un programa compuesto por una serie de herramientas que se utiliza para desarrollar código con el cual se trabajará conjuntamente con las librerías acorde a las necesidades del programa:

Entorno de Desarrollo Integrado(IDE)			
Características	Visual Studio	Eclipse	Code Blocks
Pueden crear aplicaciones que se intercomunican entre estaciones de trabajo, páginas web y dispositivos móviles	X		
Fácil de aprender.		X	X
Plataforma muy dinámica y potente		X	X
Espacios de trabajo (workspaces) para combinar múltiples proyectos	X		X
Entorno de uso más extendido, por lo que resulta fácil encontrar información, documentación y fuentes para los proyectos	X	X	
Agrega plugins y bibliotecas fácilmente		X	
Se integra con varios lenguajes informáticos	X	X	

Entorno de Desarrollo Integrado(IDE)			
Características	Visual Studio	Eclipse	Code Blocks
Entorno gráfico dinámico y en español		X	X
Software libre		X	X

Tabla 2-4 Cuadro Comparativo – Características de Lenguajes de Programación

Fuente: Autores

1) VisualStudio

Este IDE trabaja bajo el sistema operativo Windows para el desarrollo de programas en lenguajes como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo Web como ASP.NET.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios, aplicaciones Web y servicios Web en cualquier entorno que soporte la plataforma .NET.

En la figura siguiente se muestra la arquitectura de extensibilidad de Visual Studio, el IDE aloja los componentes de software llamado *VSPackages* que proporcionan una funcionalidad de la aplicación. Esta funcionalidad se comparte a través de la IDE como servicios. *VSPackages* ofrecen servicios que ellos y otros *VSPackages* usan. El IDE estándar también ofrece una amplia gama de servicios, tales como *SVsShell*, *SVsUIShell* y *SVsSolution*, que son utilizados por *VSPackages* y proporcionan acceso a la funcionalidad de ventanas IDE.

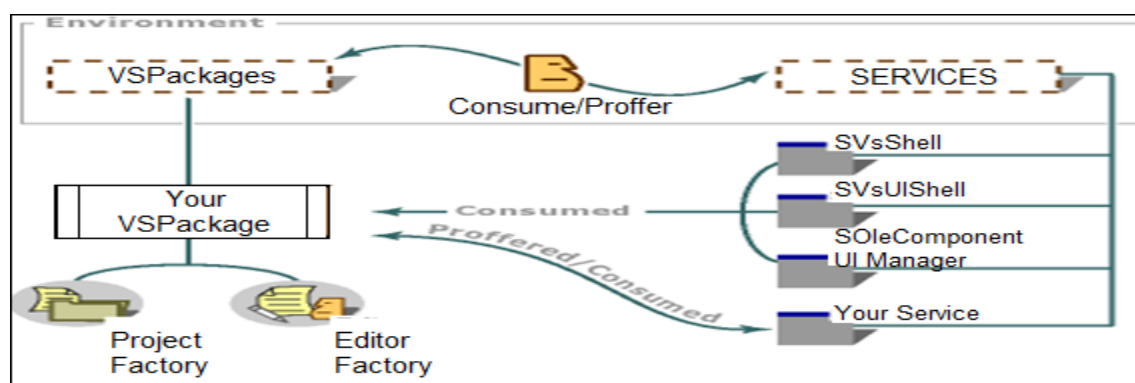


Figura 2-14 Arquitectura de Visual Studio

Fuente: Microsoft, Visual Studio Development Environment Model, 2012[43]

Características:

- La prestación de servicios críticos para su uso por *VSPackages* externos y proporciona una interfaz programable, que permite la participación de elementos de interfaz de usuario estándar.
- La creación de instancias de *VSPackages* como es requerido por las acciones del usuario o por otros *VSPackages* que solicitan servicios.
- La prestación de servicios que hacen posible la comunicación y coordinación entre *VSPackages*.
- Administra soluciones y permite coordina la selección, el contexto y la moneda.
- Proporciona la gestión de ventanas y el direccionamiento de comandos y barras de comandos, tales como menús, barras de herramientas y menús contextuales.

Características de *VSPackages*:

- Realiza la inicialización determinado y rutinas de terminación, además escribe información en el registro, que el IDE utiliza para cargar los *VSPackages* apropiadas en el momento apropiado.
- Ofrece los servicios que son necesarios para la comunicación con otros *VSPackages*.
- Proporciona implementaciones para nuevos tipos de proyectos, editores, diseñadores y extensiones para elementos de interfaz de usuario que tiene incorporados elementos de tareas, elementos de la caja de herramientas y el cuadro de diálogo de opciones.

2) Eclipse

Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar aplicaciones “cliente-liviano” basadas en navegadores[44].

La base para la arquitectura de Eclipse es la plataforma de cliente enriquecido RCP, los siguientes son los componentes que lo constituyen:

- Plataforma principal - inicio de Eclipse, ejecución de plugins

- *OSGi* - una plataforma para *bundling* estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- *JFace* - manejo de archivos, manejo de texto, editores de texto
- El *Workbench* de Eclipse - vistas, editores, perspectivas, asistentes

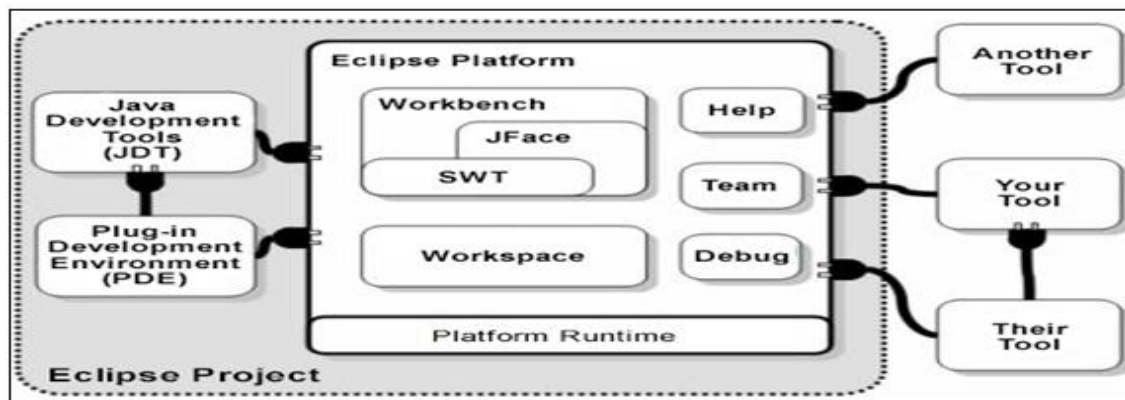


Figura 2-15 Arquitectura Eclipse

Fuente: Kehn Dan, Extend Eclipse's Java Development Tools, 2003[45]

Características

- Eclipse dispone de un editor de texto con resaltado de sintaxis y la compilación es en tiempo real.
- Tiene pruebas unitarias con *JUnit*, control de versiones con CVS, integración con ANT, asistentes (*wizards*) para creación de proyectos, clases, tests y refactorización.
- Emplea módulos (*plugin*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido y provee al programador con *frameworks* muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software y aplicaciones Web.

3) Code::Blocks

Es un IDE libre y multiplataforma para el desarrollo de programas en lenguaje C y C++, puede usarse libremente en diversos sistemas operativos.

Software para diseño 3D				
Características	3DStudio MAX	Blender	AutoCAD	Adobe Photoshop
Soporte para renderización y generación de mapeados normales	X			
Tutoriales en video y contenido online	X		X	X
Creación y Modificación de objetos 3D	X	X	X	X
Destinado a la edición, retoque y pintura a base de imágenes de mapa de bits				X
Elimina elementos de una imagen automáticamente y rellena el fondo				X
Herramienta dinámica y fácil Manejo	X		X	X
En foros se puede encontrar Información de algún requerimiento específico con facilidad	X		X	X
Mayor documentación en utilización de herramientas y modelado	X		X	X
Software propietario pero se puede utilizar demos	X		X	X

Tabla 2-5 Explicación de aplicaciones informáticas de modelado y animación 3D

Fuente: Autores

1) 3D Studio Max

Es un programa para la creación de gráficos y animación 3D, por su arquitectura basada en plugins es uno de los programas más utilizados. Con 3D Studio Max se podrá crear cualquier escena en 3D y renderizarla de una manera realista mediante el programa *mental ray*.

3D Studio Max[47] es un software que permite modelar cualquier objeto en tres dimensiones; siendo su restricción el no conocer a fondo el programa y sus diferentes características para emular elementos de la vida real como la creación y pre visualización. Es usado en diferentes campos de la vida en general como son: en la arquitectura para ver las edificaciones antes de ser construidas sobre el terreno, en el cine, en la televisión, en modelos para video juegos, en medicina para visualizar partes del cuerpo humano.

Autodesk 3ds Max soporta los siguientes formatos: max, drf, chr, fbx, 3ds, prj, ai, dae, dem, ddf, dwg, flt, htr, ige, ipt, ls, vw, lp, obj, shp, stl, trc, wrl.

Características:

- Soporta cientos de plugins para adicionar efectos fácil y rápidamente, admitiendo diferentes lenguajes de programación, permitiendo la exportación a diferentes formatos.
- Posee capacidades avanzadas de creación, modelado, animación 3D, efectos especiales, desarrollo de juegos interactivos.
- La arquitectura abierta de 3D Studio MAX permite aprovechar al máximo las herramientas y efectos disponibles para crear con facilidad cualquier efecto que se requiera.
- Muestra un rediseñado render simplificando para el proceso de creación de sorprendentes imágenes, escenas y efectos especiales realísticos.
- Se requiere de una licencia, existe la posibilidad de utilizar una versión de prueba, cuya limitación es el tiempo ya que solamente está liberada para 30 días.

2) Blender

Es un completo editor de gráficos y animaciones en 3D desarrollado en código abierto, permite crear objetos desde cero o utilizar los ya creados en el programa y combinarlos para formar otros nuevos, su interfaz es compleja, cuenta con poca información de su uso y consume muchos recursos.

La interfaz gráfica de usuario de Blender[48] es poco intuitiva, ya que no se basa en el sistema clásico de ventanas; pero tiene a su vez ventajas importantes sobre éstas, como la configuración personalizada de la distribución de los menús y vistas de cámara.

Acepta formatos gráficos como: tga, jpg, iris, sgi, o tiff. También puede leer ficheros Inventor.

Características

- Es multiplataforma, con un tamaño de origen pequeño comparado con otros paquetes de 3D, además de la edición de audio y sincronización de video.
- Contiene gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, vacíos, NURBS, *metaballs*⁹.
- Además de las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.
- Interactivo para juegos con detección de colisiones, recreaciones dinámicas y lógicas, con un renderizado interno versátil e integración externa con trazadores de rayos.
- Lenguaje Python para automatizar o controlar varias tareas y la capacidad para hacer *match moving*¹⁰.
- Motor de juegos 3D integrado, con un sistema de ladrillos lógicos y simulaciones dinámicas para *softbodies*¹¹, partículas y fluidos.
- Sistema de partículas estáticas para simular cabellos y pelajes, al que se han agregado nuevas propiedades entre las opciones de *shaders*¹² para lograr texturas realistas.

⁹Metaball: Nombre de una técnica de gráficos realizada por ordenador para simular la interacción orgánica entre diferentes objetos n-dimensionales

¹⁰Match moving: Es una técnica de efectos visuales la cual permite insertar gráficos creados en un computador en un video con la posición correcta, escala, orientación y movimiento en relación a los objetos presentes en la toma.

¹¹Softbodies: Es un campo de la infografía que se centra en simulaciones físicas visualmente realistas del movimiento y las propiedades de los objetos deformables (o cuerpos blandos).

- Licencia: software libre

Blender no posee una interfaz intuitiva con la cual se pueda conseguir resultados inmediatos sino más bien se requiere de esfuerzo para su asimilación por lo que se debe conocer primero sus conceptos para luego ponerlos en práctica.

3) AutoCAD

Permite realizar desde dibujos a mano alzada a complejas estructuras tanto en 2D como en 3D. AutoCAD utiliza una interfaz que nos permitirá insertar imágenes o mapas de bits para editarlos posteriormente con herramientas de edición gráfica que nos permitirán trabajar por capas, el programa no es nada sencillo y sin información no es posible sacar provecho. AutoCAD es uno de los mejores programas para crear planos en 2D y diseños en 3D ya que es elegido por arquitectos, ingenieros y diseñadores industriales, pero consume muchos recursos y requiere horas de aprendizaje.

Los diseños realizados en AutoCAD son de alta calidad en comparación con el dibujo tradicional a mano, pero se los debe ir guardando frecuentemente en cualquier dispositivo de almacenamiento de datos ya que si ocurre un corte de energía mientras se está creando la información se perderá.[49]

La extensión del archivo de AutoCAD es .dwg, aunque permite exportar en otros formatos como .dxf, .iges y .step para manejar compatibilidad con otros software de dibujo.

Características[50]

- Está orientado a la producción de planos, empleando para ello los recursos tradicionales de grafismo en el dibujo, como color, grosor de líneas y texturas.

¹²Shaders: Son utilizados para realizar transformaciones y crear efectos especiales como: iluminación, fuego o niebla.

- Utiliza el concepto de *espacio modelo* y *espacio papel* para separar las fases de diseño y dibujo en 2D y 3D, para obtener planos trazados en papel a su correspondiente escala.
- Un bloque es único para un dibujo específico y es posible insertar otro bloque sin que afecte al existente, además puede ser duplicado varias veces dentro de un dibujo teniendo su propia posición, escala y rotación, conocido como la creación de instancias.
- Existe la posibilidad de adjuntar un objeto en el dibujo actual como una referencia externa y si esta es actualizada la referencia externa también se actualiza para reflejar los cambios, un dibujo está compuesto de entidades como *primitivas gráficas*¹³ o *bloques*¹⁴ y los elementos gráficos son definidos geométricamente en términos del sistema de coordenadas cartesiano.
- Licencia: software propietario

4) Adobe Photoshop

Es una aplicación en forma de taller de pintura y fotografía, trabaja sobre un “lienzo” que está destinado a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits (gráficos rasterizados) y de cualquier actividad que requiera el tratamiento de imágenes digitales.

Los formatos propios de Photoshop son psd y pdd, que guardan capas, canales, guías y en cualquier modo de color, también soporta otros formatos como postscript, eps, dcs, bmp, gif, jpeg, pict, piff, png, pdf, iff, pcx, raw, tga, scitex ct, filmstrip, flashpix.

Características[51]

- Soporte de imágenes de alto rango dinámico (HDR), además posee una interfaz clásica con atajos de teclado, con un editor de gráficos vectoriales y de gráficos ráster.

¹³Primitivas gráficas: como líneas, arcos, círculos, texto

¹⁴Bloques: Es un grupo de entidades que se pueden manipular como una sola unidad

- Está escrito en C++ y soporta más de 25 idiomas muy potente incluso para uso profesional, pero debido a su gran potencia requiere una PC actualizada ya que genera un alto consumo de recursos.
- Elimina elementos de una imagen automáticamente y rellena el fondo con capas que se pueden exportar como objetos inteligentes a los que se les pueden aplicar los nuevos filtros.
- Se pueden importar objetos 3D, a los que se les puede modificar la textura desde Photoshop directamente, con una edición no-destructiva, es decir, el original se mantendrá seguro evitando el borrado o modificación accidental.
- Mejora en el soporte para vídeo: exportación compatible con formatos que adobe *premiere* y *aftereffects* pueden leer.
- Existe variedad de tutoriales en video y contenido en línea
- Licencia: software propietario

Se puede utilizar la versión de prueba de Photoshop por 30 días siendo una de sus limitaciones.

2.3.3 Dispositivo Electrónico Iteración Entorno 3D

El dispositivo electrónico con los datos que genera, puede efectuar iteraciones para manipular múltiples objetos en un ambiente virtual con modelos digitales.

La siguiente figura muestra la manera en cómo interactuará el aplicativo con el dispositivo y el entorno 3D.

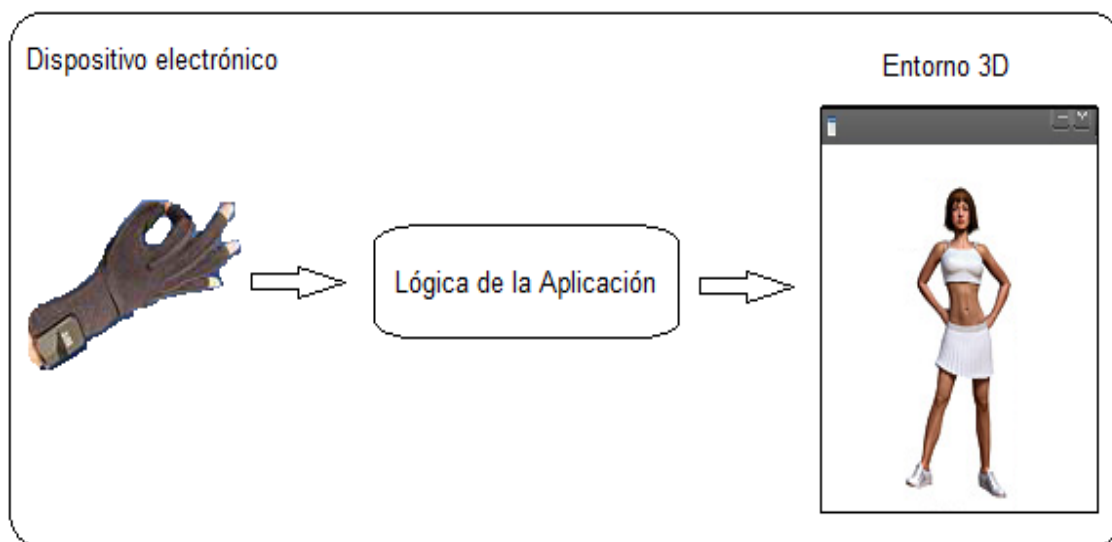


Figura 2-17 Dispositivo – Entorno 3D

Fuente: Autores

En este caso en particular se usa una interfaz gráfica sencilla para inicializar el plugin y operar el puntero del mouse para luego manipular el entorno 3D.

Mediante la instalación del conjunto de bibliotecas listado en el Anexo Librerías y paquetes de instalación C++ o a través del uso de un *script* con los archivos necesarios citados en dicho anexo, se podría hacer la recepción de datos mediante el puerto USB.

2.3.4 Clúster

Es un conjunto de computadoras interconectadas con dispositivos de alta velocidad que actúan en conjunto usando el poder de cómputo de varias CPUs en combinación para resolver ciertos problemas dados.

La ventaja de trabajar con Clústers es la de construir una plataforma la cual se ajuste a un presupuesto determinado y que sea apropiada para un grupo considerable de aplicaciones como algoritmos genéticos, simulación de línea de fabricación, aplicaciones militares, base de datos, síntesis de imágenes, recuperación de imágenes por contenido, simulación de modelos para clima, análisis de sismos, algoritmos para solución a problemas de electromagnetismo, dinámica de fluidos,

química cuántica, biomedicina; siendo expandido con gran facilidad incrementando un número de nodos o añadiendo memoria o procesadores[52].

Se encuentran disponibles varias herramientas para la administración y manejo del Clúster, tanto en herramientas de monitoreo, así como de herramientas para la administración de trabajo y recursos.

Características especiales:

- Un Clúster consta de 2 o más nodos y los nodos están conectados entre sí por al menos un canal de comunicación.
- Los Clústers necesitan un software de control especializado.

Clasificación

El Clúster tiene diferentes connotaciones en base al uso que se dé y a los servicios que ofrece por lo que el significado de un término cambiará para el grupo que lo utilice:

- a) **High Performance:** Para tareas que requieren gran poder computacional, grandes cantidades de memoria, o ambos a la vez y las tareas podrían comprometer los recursos por largos periodos de tiempo, por ejemplo para un grupo involucrado en computación científica.
- b) **High Availability:** Máxima disponibilidad de servicios y rendimiento sostenido, por ejemplo para servicios Web disponibles incluso frente a fallas.
- c) **High Throughput:** Independencia de datos entre las tareas individuales, el retardo entre los nodos del Clúster no es considerado un gran problema y la meta es el completar el mayor número de tareas en el tiempo más corto posible, por ejemplo para grupos cuyo interés está en los Clústers que permiten ejecutar un gran número de tareas independientes en paralelo.

En la siguiente tabla se realizará la comparación de las características entre algunos sistemas de la alta disponibilidad y escalabilidad de Clústeres Linux para luego seleccionar el que mejor se adapte a las necesidades del aplicativo:

Clúster			
Características	OSCAR	ROCKS	OpenMosix
Compatibilidad con varias distribuciones de Linux		X	
Se utiliza para la computación científica mediante MPI	X	X	X
Reduce la necesidad de expertos en la creación de un clúster.	X	X	X
Ampliamente utilizado en la computación de alto rendimiento	X	X	X
Integra una distribución validada con una solución confiable y repetible		X	
Los monitores y los teclados son útiles, pero no son necesarios	X	X	X
Soporte en librerías para programación en paralelo		X	
No es sencilla la configuración de los paquetes instalados	X		X
Ausencia de herramientas adecuadas para automatizar la instalación de todos los nodos			X
Sistema fácil de usar, actualizar y mantener.		X	
Incluye un completo conjunto de herramientas llamadas <i>rolls</i> ¹⁵		X	
Mayor información y documentación ya que cuenta con una comunidad activa, con diferentes foros y listas de discusión para la solución de diversos problemas		X	
Actualmente el desarrollo del sistema del Clúster está parado.			X

Tabla 2-6 Características de los clúster OSCAR, ROCKS y OpenMosix

Fuente: Autores

¹⁵Rolls: Son paquetes con herramientas de diversa funcionalidad orientados hacia un problema concreto

1) OSCAR

Incluye una arquitectura robusta y extensible, se utiliza generalmente para la computación científica mediante una interfaz de paso de mensajes (MPI), varias de las cuales están incluidas en el paquete de OSCAR[53].

La Figura 2-18 ilustra la arquitectura de HA-OSCAR:

El servidor primario *standby* es el responsable de recibir y distribuir las solicitudes de los clientes, activar sus servicios, monitorear sus funciones y cuando haya un fallo en el mismo lo detectará.

Los *switches* LAN locales proporcionan conectividad local entre la cabeza y el cliente / nodos de computación.

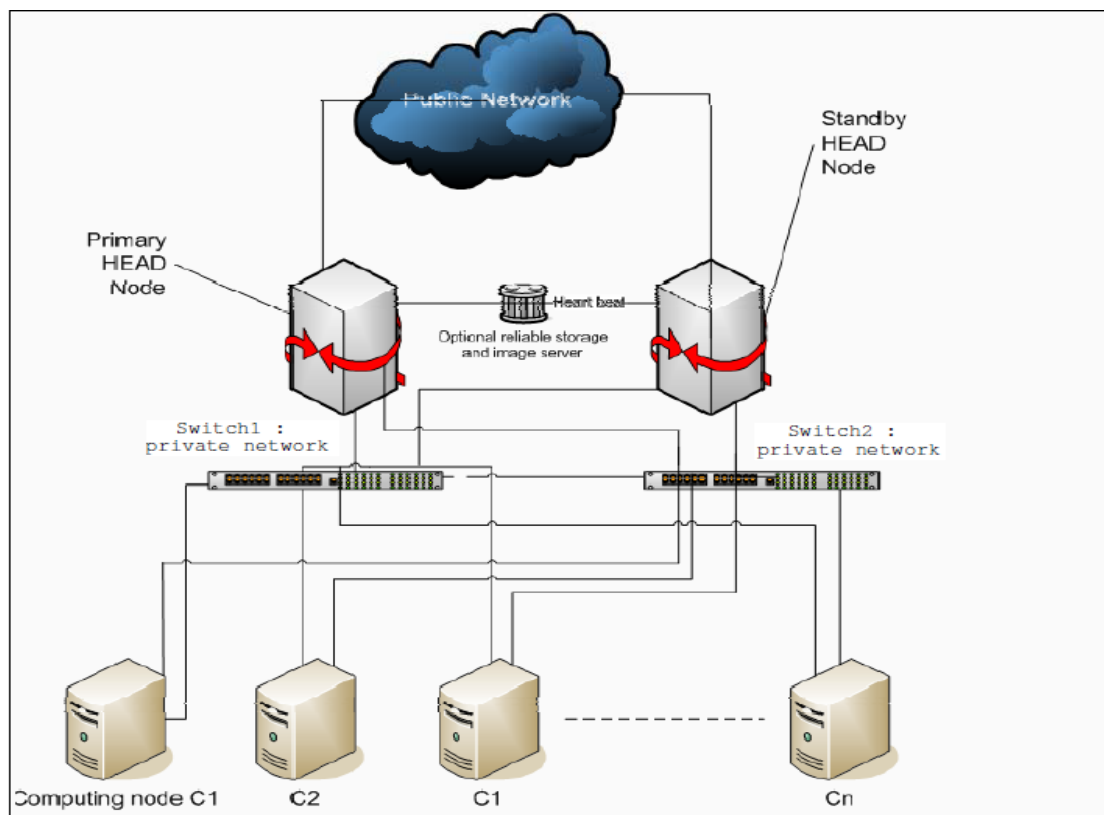


Figura 2-18 Arquitectura de Clúster HA-OSCAR

Fuente: Sloan Joseph D., High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI: With OSCAR, Rocks, openMosix, and MPI, 2004[54]

Características:

- Oscar ha sido probado para trabajar con una distribución de Linux, aunque tiene algunos problemas de compatibilidad con algunas de las distribuciones...
- Oscar incluye varias características en las áreas de disponibilidad, escalabilidad y seguridad; redundancia en el nodo cabeza, auto-recuperación para el hardware, el servicio y las interrupciones de la aplicación.
- Los monitores y teclados son útiles, pero no necesarios y debido a la variedad de combinaciones de software y hardware no se puede saber el tipo de hardware que soporta, si el hardware funciona en una instalación de archivo del sistema operativo existe la probabilidad de que no habrá problemas con OSCAR.

2) ROCKS

Rocks proporciona toda la capacidad necesaria para implementar un Clúster para el análisis de gran cantidad de datos en computación de alto rendimiento.

Características:

- Rocks es la pila completa de software de gran infraestructura integrada que incluye un completo conjunto de instrucciones útiles para la administración.
- Se pueden construir software de acuerdo a los requerimientos del usuario, seleccionando que complementos van incluidos en el diseño, eligiendo entre los distintos *Rolls* para los sistemas operativos (Red HatEnterpriseLinux, Linux Oracle o CentOS).
- Es el estándar más usado en *HPCMiddleware*¹⁶ siendo la forma más factible de construir y mantener un clúster.
- Tiene soporte en librerías para la programación en paralelo, permitiendo que nuevas aplicaciones consigan funcionar con un multiprocesamiento más sencillo.
- Viene con soporte de clase empresarial comercial de StackIQ[55], líderes de la comunidad Rocks.

¹⁶Middleware: Es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

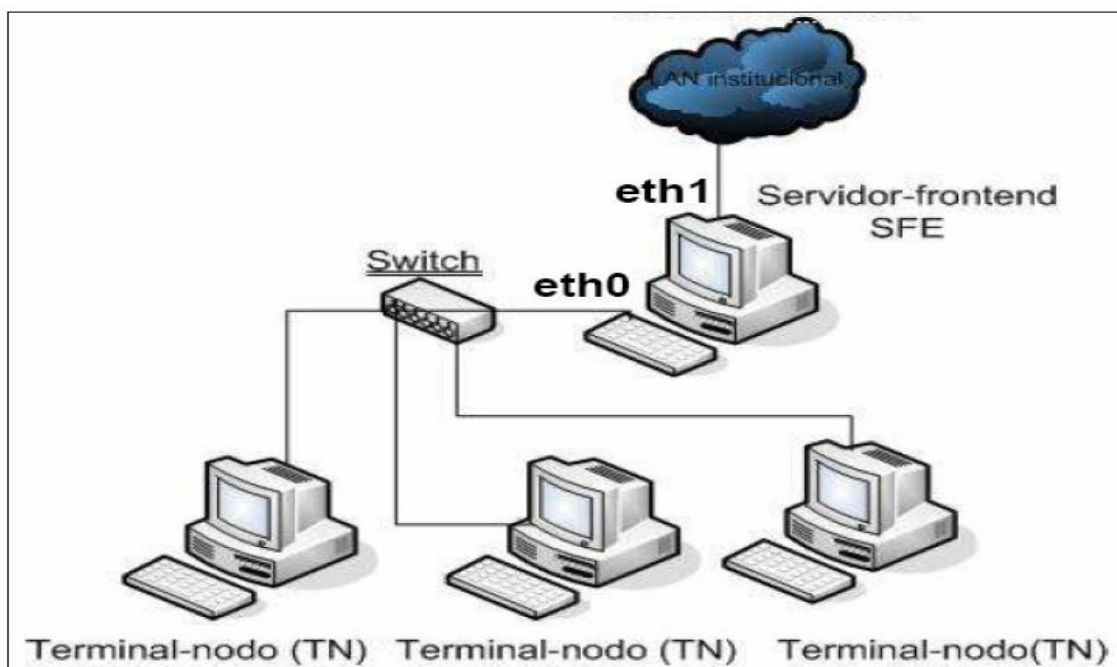


Figura 2-19 Esquema general de configuración física de un Clúster

Fuente: Guerrero Adan, Instalación y configuración de un clúster con Rocks en la Universidad de Guadalajara, 2009[56]

El clúster es escalable y cuenta con la facilidad de extenderlo con equipos a bajo costo y no representa una limitante al momento de agregar recursos necesarios para incrementar el poder de cómputo.

3) OpenMosix

Es un sistema de Clúster SSI(Single SystemImage), que añade funcionalidades al kernel del Sistema Operativo para el agrupamiento de una sola imagen, ofreciendo a usuarios y a aplicaciones la ilusión de un solo equipo con varias CPUs, conveniente para distribuir cálculos entre máquinas de propósito general.[57]

Características:

- No requiere de paquetes extras y no es necesario modificaciones en el código, es escalable y adaptable permitiendo a varias máquinas actuar como un único sistema multiprocesador¹⁷.

¹⁷Multiprocesador: Computador que cuenta con dos o más microprocesadores (CPUs)

- Es dependiente del kernel; permite la adición o sustracción de nodos en caliente sin necesidad de interrumpir el servicio.
- No migra todos los procesos siempre, tiene limitaciones de funcionamiento y problemas con memoria compartida y los procesos con múltiples hilos no tienen mucha eficiencia, tampoco obtiene una mejora cuando se ejecuta un solo proceso, por ejemplo un navegador Web.

Implementa un algoritmo balanceador de carga¹⁸ permitiendo repartir de forma óptima la carga entre los diferentes nodos, si está correctamente calibrado. Dado que la migración de procesos se hace de forma transparente el conjunto de Clúster se muestra como un gran sistema multiprocesador con tantos procesadores disponibles como la suma de los procesadores de todos los nodos.

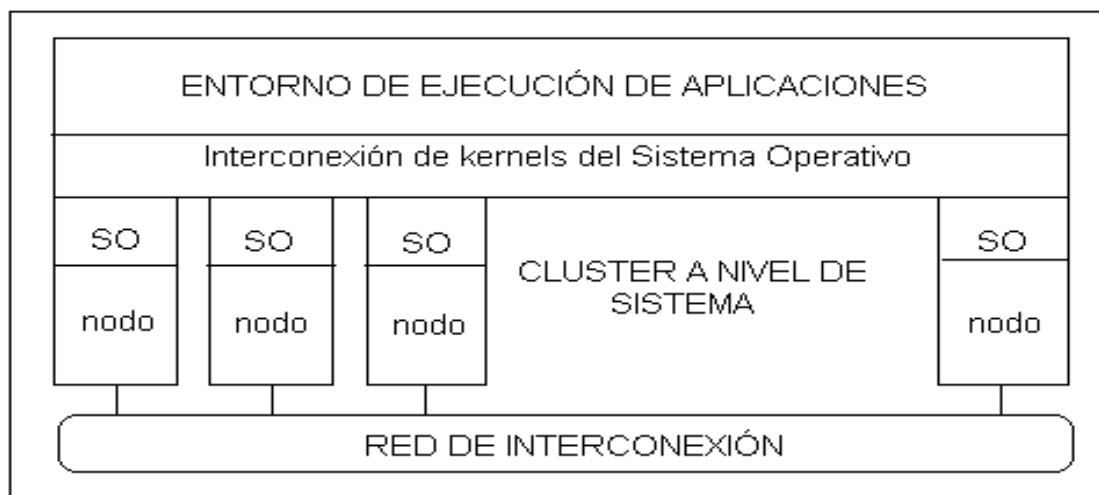


Figura 2-20 Arquitectura de un Clúster tipo SSI[58]

Fuente: OpenMosix, Sistema de computación masiva en Sun Grid, 2010

¹⁸Balanceo de carga: Cuando los recursos del nodo son insuficientes para el procesamiento de datos, el Clúster distribuye entre los demás nodos las tareas para un mejor desempeño.

CAPÍTULO III

3. SELECCIÓN DE HERRAMIENTAS PARA ENTORNO 3D

3.1 HTML

Se ha convertido en el formato más fácil para la creación de páginas Web debido a su sencillez, por ser extensible, y permitir el uso de Applets. Se pueden añadir características, etiquetas y funciones adicionales para el diseño Web, generando un producto vistoso y rápido, por ello se lo ha seleccionado para ser utilizado como el lenguaje que contendrá al aplicativo del proyecto. Permite describir hipertexto y no necesita de grandes conocimientos cuando se cuenta con un editor de páginas Web. El texto es presentado de forma estructurada y agradable en archivos pequeños con un despliegue rápido. Es un lenguaje de fácil aprendizaje y lo admiten todos los exploradores.

HTML tiene ciertas limitaciones ya que es un lenguaje estático y cuenta con etiquetas muy limitadas para su desarrollo y la interpretación de cada navegador puede ser diferente, ya que guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.

Está diseñado para usarse sobre Internet, intranets y sistemas locales. El cual puede transmitirse e inter-relacionarse a través del Web y visualizarse mediante algún navegador.

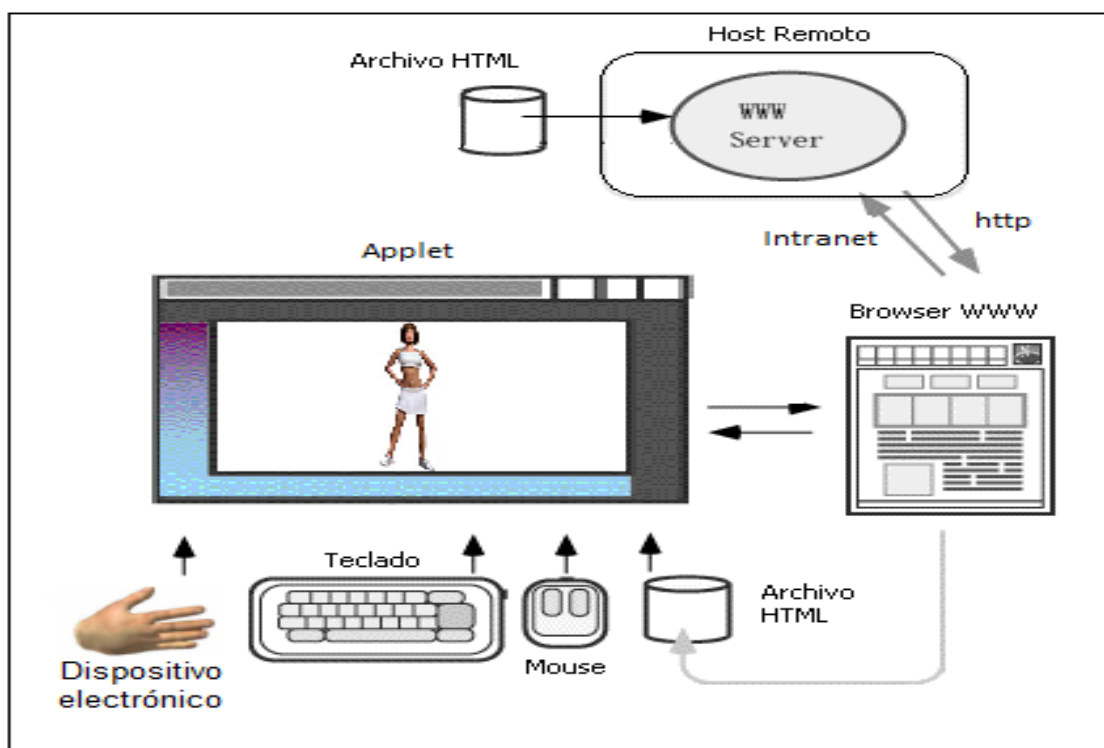


Figura 3-1 Arquitectura HTML – Dispositivo Electrónico – Aplicativo 3D

Fuente: Autores

Los Applets son programas escritos en lenguaje Java, diseñados para ser ejecutados desde un navegador, que se puede colocar en un servidor, junto con el resto de ficheros que componen un sitio Web.

Los atributos que el *tag* usará para ubicar al Applet son los siguientes:

Atributos de un Tag	
Atributo	Característica
CODE	Para ejecutar el Applet, es el nombre del fichero "class".
CODEBASE	Directorio del Applet "class".
WIDTH	Ancho del Applet.
HEIGHT	Alto del Applet.
ALIGN	Alineación del Applet para texto circundante.
VSPACE	Espacio vertical dejado como margen entre el Applet y el texto.
HSPACE	Espacio horizontal dejado en blanco como margen entre el Applet y texto circundante.
BORDER	Espacio circundante vacío del Applet.

Atributos de un Tag	
Atributo	Característica
NAME	Nombre del Applet. Para hacer referencia al Applet desde otro o desde javascript.
ARCHIVE	Una colección comprimida de los componentes del Applet.
MAYSCRIPT	Indica si el Applet de Java puede acceder a los objetos de secuencias de comandos de la página Web.
PARAM NAME	"Nombreparámetro" VALUE="valor parámetro".

Tabla 3-1 Cuadro de los atributos que un *tag* utiliza

Fuente: Autores

A continuación se muestra los pasos que el navegador sigue para la ejecución de un Applet[59].

- Se realiza la carga del URL en el navegador
- El navegador carga el documento de HTML
- Las clases de Applet son cargadas por el navegador
- El navegador ejecuta el Applet

Para ponerlo en marcha se necesita un archivo HTML que incluya el siguiente *tag* en su interior:

```
<applet
  code=class-filename
  width=pixels
  height=pixels
  archive=jar-file
  <paramname=parameter1value=value1>
</applet>
```

3.2 C++

C, C++ son lenguajes de programación procedural y orientado a objetos, versátiles, generales y con potencia a bajo nivel, se integran de manera ágil con el hardware además de ser sencillos y robustos, C++ es el mejor al momento de usarlo con

interfaces pequeñas y según los cuadros comparativos es uno de los dominantes por lo cual se lo ha elegido para el módulo de integración con el dispositivo electrónico así como para el aplicativo 3D.

Características:

- La programación es procedural, orientada a objetos, potente, expresiva, eficaz y de alto nivel, siendo rápida y sencilla para el desarrollo de proyectos pequeños.
- Consume menos memoria RAM por ser un lenguaje compilado directamente en código máquina.

A continuación se pueden apreciar algunas de las ventajas que proporciona este lenguaje así como sus desventajas.

Ventajas:

- Esta enriquecido con clases que describen tipos de datos adaptados a diferentes necesidades para el programador, formado por instrucciones muy explícitas, cortas, cuya duración de ejecución puede preverse al momento de escribir el programa.
- Los programas construidos describen tipos estructurados, variables y procedimientos (funciones) y sirve para escribir sistemas operativos o partes de los mismos como el sistema en que están dotados los Macintosh, así como la interfaz gráfica de Windows.
- Los encabezamientos propios al programa definen constantes, funciones y clases que se implementan en archivos .cpp

Desventajas:[60]

- En un entorno de ejecución como internet el ejecutable C++ constituye un riesgo porque no se puede predecir a simple vista si un archivo ejecutable contiene instrucciones peligrosas como por ejemplo formatear el disco duro.
- Es descrito por una gramática sensible al contexto, por lo que el algoritmo tiene que analizar la alta complejidad.

- Utiliza los archivos de cabecera como C y el tiempo de compilación depende del número de tipos de trabajos que se ejecuten

A continuación se describen algunas librerías que serán utilizadas para la implementación de la interacción dispositivo electrónico – aplicativo 3D:

Librerías: Las librerías utilizadas para la comunicación con el dispositivo electrónico son: Libserial y X11 junto con su interfaz gráfica, por lo que a se describen continuación:

- **Libserial.** Proporciona una serie de clases en C++ para acceder al puerto *serie* como objetos de entrada/salida y *serialport*. Con la función miembro de esta librería se ajustan diversos parámetros del puerto *serie* como: velocidad de transmisión, tamaño de la fuente, el control de flujo, entre otros.
- **X11 (XWindow).** *Xlib* es una librería de bajo nivel que contiene un grupo de funciones hechas en C de código abierto para el uso de la interfaz gráfica del servidor X11, en conjunto con la biblioteca *Motif*, se los usa a manera de cliente permitiendo manipular ventanas, botones y primitivas gráficas¹⁹.

3.3 Java

La elección de éste lenguaje de programación es debido a que es un software libre, es decir, no hay la necesidad de comprar licencias para su utilización, además es rápido, seguro y confiable. Por lo que se eligió para desarrollar el aplicativo 3D de acuerdo al cuadro comparativo descrito anteriormente (Ver Tabla 2-3).

Características:

- Es robusto por ser orientado a objetos permitiendo interactuar y enlazar varios sistemas, además hace aplicaciones modulares, es decir, divide el código fuente en varios ficheros para una mejor organización y poder conocer donde esta cada cosa.

¹⁹ Primitivas Gráficas: se refiere a Puntos, Rectas, Circunferencias, Polígonos.

- Muy bueno para proyectos grandes o aquellos en donde se utiliza mucho código y suministra bibliotecas adicionales para acceder a las características de cada dispositivo como: gráficos, ejecución mediante hebras o hilos, interfaz de red de forma única.

Los hilos son acciones paralelas que pueden ejecutar un programa al mismo tiempo, es decir, un hilo puede realizar un cálculo usando el mismo tiempo que otro hilo que permita una interacción del usuario, por tal motivo la espera será menor. Java tiene propiedades de sincronización haciendo de este tipo de entornos de programación más fácil.

A continuación se pueden apreciar algunas de las ventajas que proporciona este lenguaje así como sus desventajas[61].

Ventajas:

- Es independiente de la plataforma en la que se ejecuta el programa pudiendo ejecutarse en cualquier tipo de hardware, ya que Java *codees* un código de bytes y no código máquina, por lo que requiere de un procesador virtual Java que interprete y ejecute el código de bytes.
- Posee gran similitud con el lenguaje de programación C++ pero no tiene punteros y todas las entidades de Java son objetos, además la interfaz permite al programador casi todo lo que se puede conseguir con herencia.
- Los Applets son fáciles de integrar en páginas Web, pueden ser descargados de la Web y de gestión utiliza los recursos del equipo local que ejecuta programas Java.

Desventajas:

- El navegador de Internet Explorer no admite la ejecución de Applets sin un conector (plugin) aparte.

- El rendimiento se reduce ya que el código Java es descifrado por el intérprete, por lo que las aplicaciones Java son más lentas que las aplicaciones en código nativo en OpenGL o DirectX escritas en C++.

Librería de modelado y gráficos 3D OpenGL

La librería para pegar OpenGL y el sistema de ventanas depende del sistema elegido siendo, posible encontrar librerías que encapsulan el trabajo con el API del entorno de ventanas y que facilitan la creación y gestión de ventanas como GLUT cuyas funciones llevan el prefijo *glut*-.

GLUT es un API con formato ANSI C para escribir programas OpenGL independiente del sistema de ventanas utilizado. GLUT es una librería pequeña, fácil de aprender y utilizar, siendo el API de GLUT una máquina de estados donde el sistema tiene un conjunto de variables de estado que durante la ejecución del programa fijan las características que debe tener la aplicación en cada momento.

Las funciones de GLUT son simples, con pocos parámetros y evitan el uso de punteros. Dependiendo de la funcionalidad podemos distinguir funciones para: inicialización, control de ventanas, control de menús, procesamiento de eventos, registro de funciones *callback*, obtención del estado, control de *overlays*, control del mapa de colores, visualización de fuentes de texto y dibujo de formas geométricas

Varias bibliotecas se construyen encima o al lado de OpenGL para proporcionar características no disponibles en OpenGL, habiendo bibliotecas de más alto nivel orientado a objetos para un escenario gráfico.

Sistema de coordenadas

Para que la visualización de los objetos tridimensionales se realice se creará los sistemas de coordenadas adecuados.

El mundo real es la región del espacio cartesiano 3D en el cual se encuentran los objetos que se van a visualizar, midiéndose en coordenadas continuas y de longitud

y para que los objetos puedan ser visualizados sobre una pantalla en dos dimensiones deben soportar diferentes transformaciones.

Los objetos del mundo real son vistos desde un punto del espacio, llamado punto de vista²⁰. Este punto define un nuevo sistema de coordenadas, denominadas coordenadas del ojo que es donde se encuentra el ojo que mira al mundo real. Desde el momento en el cual se especifican las coordenadas de un elemento de una escena, hasta que se realiza la representación final en la pantalla, OpenGL realiza cinco transformaciones que se describen a continuación.

La transformación de proyección. Especifica cómo se tiene que ver la imagen en la pantalla (volumen de visualización de la escena) limitada en ancho, alto y profundidad, para poder visualizar los objetos en pantalla es necesario transformar las coordenadas tridimensionales (x,y,z) en puntos del espacio 2D (x,y) utilizando alguna proyección (ortográfica²¹ o perspectiva²²) y teniendo en cuenta el orden de aplicación de las transformaciones. Esta transformación se realizará sobre la orientación que se haya dado hasta el momento al modelador.

La transformación de Viewport. Ajusta el producto de la proyección a las dimensiones de un rectángulo contenedor (ventana).

La transformación de vista. Consiste en poner los puntos del objeto en función del sistema de coordenadas del ojo.

La transformación de modelo. Se aplica sobre los objetos que componen una escena; en este punto se puede aplicar cualquier transformación (escalado, rotación y translación) sobre los objetos de una escena.

La transformación de Modelo-Vista. Es la combinación de las dos transformaciones anteriores, que desde un punto de vista práctico son semejantes.

²⁰Punto de vista. Es el punto del espacio en el que está situado el observador

²¹Proyección ortogonal: La imagen final contiene los objetos con las mismas dimensiones con las que se encuentran especificados en la escena.

²²Proyección en perspectiva: Este tipo de proyección juega con el tamaño de los objetos con el fin de crear una escena lo más realista posible, de esta manera se crea una ilusión de profundidad sobre una superficie 2D.

Los pasos que se deben dar para convertir las coordenadas tridimensionales de un objeto en el mundo real a coordenadas bidimensionales discretas de la ventana (en pixels) son como muestra la Figura 3-2.

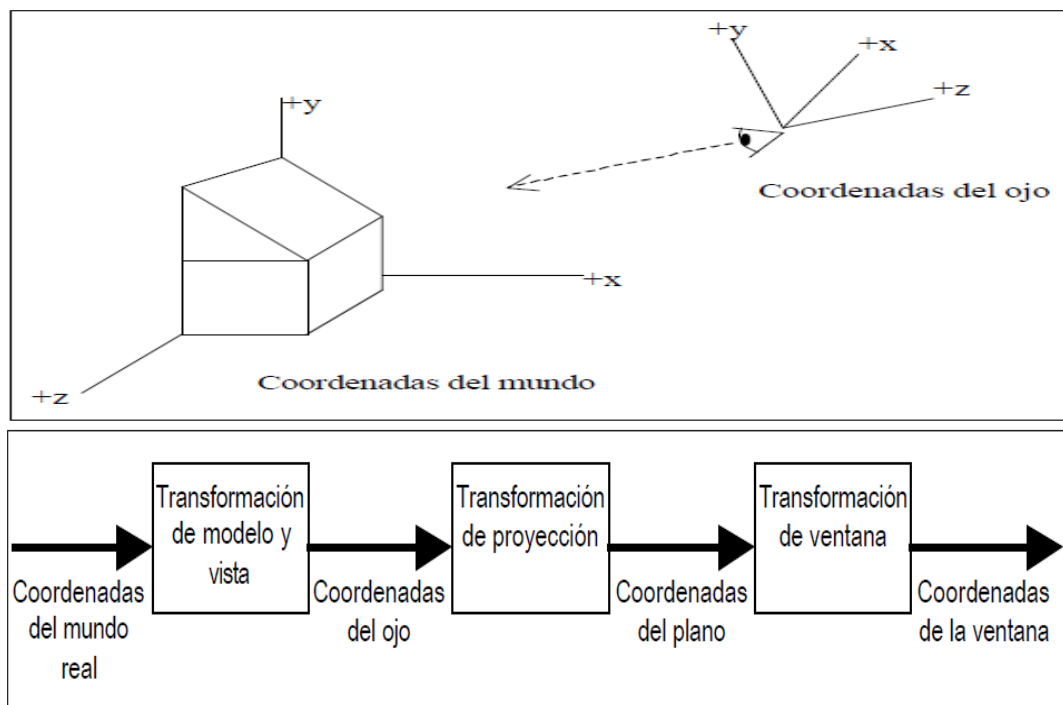


Figura 3-2 Coordenadas del mundo real y secuencia de transformación de vértices

Fuente: Molina Carmona y Puchol García, APUNTES DE OPENGL, 1999[62]

OpenGL usa el sistema de coordenadas dextrógiro²³, es decir, que en el eje vertical esta la Y positivo, en el eje horizontal la X positivo y la Z positiva saliendo de la pantalla hacia el espectador, no siendo así en Direct3D que usa el sistema levógiro²⁴.

El sentido de giro se define mediante lo siguiente:

Giro según el eje x – La dirección positiva de rotación va del eje y al eje z.

Giro según el eje y – La dirección positiva de rotación va del eje z al eje x.

Giro según el eje z – La dirección positiva de rotación va del eje x al eje y.

²³ Dextrógiro: Cuando gira en el mismo sentido que las agujas del reloj

²⁴ Levógiro: Cuando gira en sentido contrario a las agujas del reloj

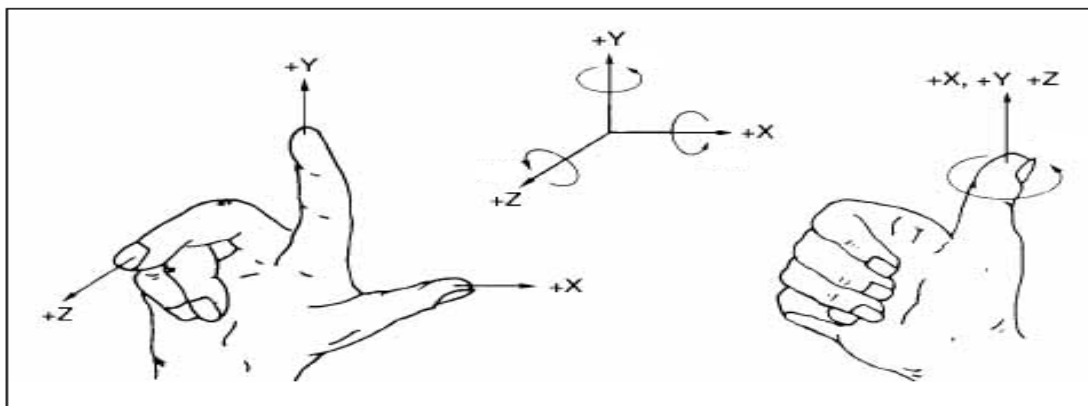


Figura 3-3 Coordenada dextrógiro

Fuente: Fagor Automation, Nomenclatura de los ejes en CNC, 2012

OpenGL proporciona funciones para controlar la traslación, rotación y escalado, estas transformaciones se representan como matrices 4x4 ordenadas como vectores columna.

Matrices de transformación:

Se pueden aplicar tres operaciones básicas a los vértices de un objeto: traslación (reposiciona un objeto desplazándolo de una posición a otras nuevas coordenadas), rotación (los puntos sobre los que se aplica desplazarán y rotarán respecto a un punto determinado) y escalado (aumenta o disminuye las dimensiones de una imagen). Para aplicar una de estas operaciones basta con multiplicar el punto en coordenadas homogéneas por una matriz y estas son:

Traslación				Escalado			
$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$				$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$			
Rotación							
$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$		$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$		$R_z(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$			

Figura 3-4 Coordenadas del mundo real y secuencia de transformación de vértices

Fuente: Molina Carmona y Puchol García, APUNTES DE OPENGL, 1999[62]

Se debe tener en cuenta el orden de aplicación de las transformaciones en OpenGL ya que no son conmutativas²⁵ entre sí.

3.4 Biblioteca Gráfica Java 3D

Java 3D ha sido desarrollada bajo la colaboración conjunta de las empresas Intel, Silicon Graphics, Apple y Sun, que combinando sus conocimientos han diseñado un API independiente de la plataforma sobre el Sistema Operativo anfitrión como: PC, Solaris, Irix, HP-UX, Linux y la plataforma de gráficos: OpenGL, Direct3D, así como los dispositivos de entrada y de salida (display) [63].

El API Java 3D es una adición a Java que se utiliza para la visualización de gráficos en tres dimensiones, con una gran colección de constructores y funciones para crear y manipular objetos geométricos 3D los cuales residen en un universo virtual. Además los programas escritos en Java 3D pueden ejecutarse en diferentes tipos de equipos y en Internet.

Java 3D permite a los programadores la posibilidad de manipular geometrías complejas en tres dimensiones y crear aplicaciones gráficas 3D ya que por su versatilidad soporta un gran número de formatos como VRML.

Debido a que los sistemas de información gráfica en 3D permiten manipular la ubicación de los objetos en una escena, se debe poner en práctica la estructura de datos más eficiente junto con el renderizado para representar la geometría de una escena.

²⁵Propiedad conmutativa: Es cuando el resultado de una operación es el mismo cualquiera que sea el orden de los elementos con los que se opera.

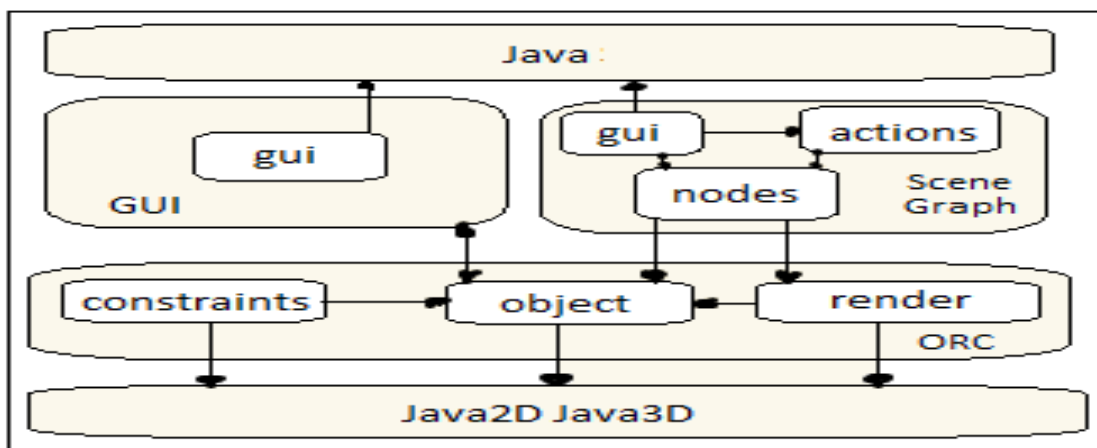


Figura 3-5 Arquitectura Java 3D

Fuente: Hanisch Frank, European Journal of Open, Distance and E-Learning, 2000[64]

- Como visión general de la arquitectura Java 3D se puede decir que tiene componentes independientes asíncronos y que está diseñada para el paralelismo. Tiene comportamiento y programación de sonido así como generación de eventos (detección de colisiones) y entrada de gestión de dispositivos, Java 3D render elige un orden transversal.

Características:

- Tiene capacidad suficiente para producir juegos y animación, de los objetos se puede controlar tamaño, posición, orientación y dichos atributos evolucionan en el tiempo.[65]
- Posee una interfaz sencilla que la mayoría de las bibliotecas de otros gráficos pero los detalles de implementación internos de Java Sun 3D están sujetos a cambio en cualquier momento.
- Se basa en la tecnología existente, como DirectX y OpenGL por lo que los programas no se ejecutan tan lentamente como se podría esperar.
- Orientada a objetos y permite un alto nivel de desarrollo de aplicaciones Java y Applets.

A continuación se pueden apreciar algunas de las ventajas que proporciona esta biblioteca así como sus desventajas.

Ventajas:

- Java 3D optimiza el renderizado (proceso de generar una imagen desde un modelo), permitiendo manejar automáticamente lo más rápido posible al ordenador la escena en tres dimensiones.
- Puede incorporar objetos creados por paquetes de modelado 3D como *truespace* y modelos VRML.

Desventajas:

- Uno de los problemas de Java 3D es que crea objetos los cuales posteriormente se convertirán en basura y los procesos del recolector de basura de Java 3D empezarán a ejecutarse, ocasionando una baja en el desempeño del renderizado perdiendo realismo en la escena.
- Java 3D es una extensión que no forma parte de la distribución estándar de Java y la programación en tres dimensiones puede ser complicada debido a la cantidad de tecnicismos y a las matemáticas involucradas.

El sistema de coordenadas del universo virtual J3D es de mano derecha como indica la Figura 3-3.

Para la instalación y ejecución de Java 3D y el desarrollo del aplicativo previamente se tiene que configurar los paquetes necesarios para el correcto funcionamiento.

La versión a configurar del JRE (ver Anexo Configuración de JRE), JDK (ver Anexo Configuración de JDK) y J3D (ver Anexo Configuración Java 3D - API 1.4.0_01) se lo hará de forma manual, para saber la localización en la que se encuentra y poder enlazar las librerías donde se requiera para que se ejecute el aplicativo en Java y en el navegador Web.

En la Figura 3-6 se puede apreciar la interfaz gráfica y el entorno ejecutado de Java 3D.

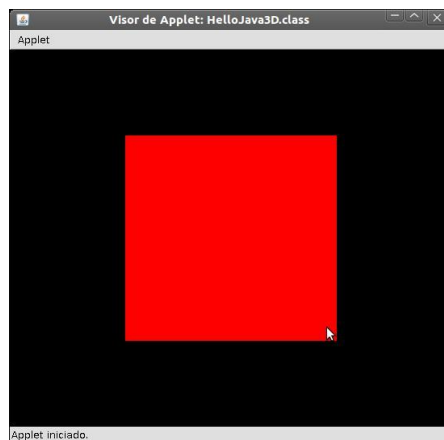


Figura 3-6 Applet de Hola Mundo en Java 3D

Fuente: Autores

3.5 Biblioteca Gráfica OpenSceneGraph

OpenSceneGraph es un “*toolkit*” gráfico de alto nivel; portable para el desarrollo de aplicaciones gráficas de alto rendimiento. Es útil para la visualización científica ya que se fundamenta en el concepto de grafo de escena que provee de un entorno de trabajo orientado a objetos, añadiendo una multitud de beneficios para el rápido desarrollo de aplicaciones gráficas ya que está construido a partir de la librería gráfica OpenGL, esto libera al desarrollador de implementar y optimizar llamadas gráficas de bajo nivel, y provee muchas ventajas adicionales para un rápido desarrollo de aplicaciones gráficas.[36]

Características

- Está escrito completamente en C++ y OpenGL, aplicando útiles patrones de diseño de software para estructurar la biblioteca.
- Como parte del núcleo del grafo de escena, soporta ordenación de estados, arreglos de vértices y listas de despliegue, lo cual da mejoramiento de función a los *grafos* 3D, además se puede personalizar el proceso de dibujo para añadir características que mejoren el desempeño.
- OSG encapsula en clases casi todo OpenGL y sus extensiones, sin tener que preocuparse en la codificación a bajo nivel. Es extensible, por lo que los desarrolladores pueden integrarlo fácilmente con sus aplicaciones.

- Es independiente de la plataforma, el núcleo de OSG requiere tan solo de C++ y OpenGL, de modo que funciona en: IRIX, Linux, Windows, FreeBSD, Mac OSX, Solaris, HP-UX y hasta PlayStation2.
- Soporta desde un sólo despliegue hasta múltiples despliegues, como los requeridos en las instalaciones para realidad virtual.
- Posee una comunidad de usuarios/desarrolladores de OSG muy activa, por lo que se puede conseguir ayuda en sus listas de correos, aunque casi toda la información está en inglés. Existen constantes contribuciones en el desarrollo de OSG y en el desarrollo de bibliotecas externas asociadas con OSG, además hay un gran número de aplicaciones Open Source (código abierto) y comerciales creadas con OSG para diversos propósitos.

Por éstas características y más particularidades, se ha elegido OSG para realizar el módulo de manipulación de imágenes 3D.

Formatos soportados por OSG[36]

OpenSceneGraph	
Formato	Descripción
.osg, .ive	Nativo de OSG, en texto plano y binario respectivamente.
.3dc	Vista en 3D de edificios y casas.
.3ds	3D Studio Max, programa de modelado y renderizado gráfico.
.ac	AC3D.el formato de archivo de texto ASCII.
.dw	Designer Workshop.
.dxf	Autocad, crea planos en 2D y diseños en 3D
.flt	OpenFlight, formato de grabación de escenarios en 3D.
.geo	Carbon Graphics.archivo que se utiliza para georreferenciar imágenes en DigiNG
.iv, .wrl	Inventor, permite diseñar e implementar aplicaciones
.logo	Golden Orchard de Apple II.
.lwo, .lws	Lightwave, programa de animación y creación

OpenSceneGraph	
Formato	Descripción
	de objetos y espacios en tres dimensiones.
.md2	Quake2, formato de archivo diseñado para guardar modelos 3D, con compresión y pérdida de datos.
.obj	Wavefront, Formato de archivo 3D
.pfb	Performer, Fuente PostScript en formato binario
.x	Direct3D de DirectX.

Tabla 3-2 Formatos de archivo soportados por OSG

Fuente: Autores

A continuación se pueden apreciar algunas de las ventajas que proporciona esta biblioteca así como sus desventajas.

Ventajas:[66]

- Da mayor realismo 3D que otras bibliotecas con una codificación libre la cual es accesible y clara con la finalidad de buscar la portabilidad de las bibliotecas en los diferentes sistemas.
- Cuenta con diversas formas de navegación con las cuales el usuario puede mover los objetos virtuales, inspeccionarlos o moverlos a través de la escena.
- Posee animación de objetos y los cargadores de escenas 3D permiten leer varios archivos en diversos formatos.
- La integración de interfaces de interacción con las escenas 3D, existen diversos dispositivos que permiten manipular los objetos virtuales. Por ejemplo, dispositivo electrónico para “tomar” objetos virtuales o los complejos brazos electro-mecánicos para la simulación de cirugías.

Desventajas:

- Opciones de desarrollo comercial con software como *Virtools* y *Quest3D* que permite elaborar aplicaciones por medio de programación visual que funcionan solo en la plataforma Windows siendo herramientas de costos muy elevados.

- Tiene problemas de compatibilidad, estabilidad y fallos en CentOS y Windows (especialmente en versiones anteriores)

Para el desarrollo del aplicativo se requiere la instalación de los paquetes OpenSceneGraph-3.0.1.zip y OpenSceneGraph-Data-3.0.0.zip (Anexo Instalación de OSG) para el correcto funcionamiento.

En la interfaz de la Figura3-7 se puede apreciar la compilación del hola mundo de la biblioteca OSG

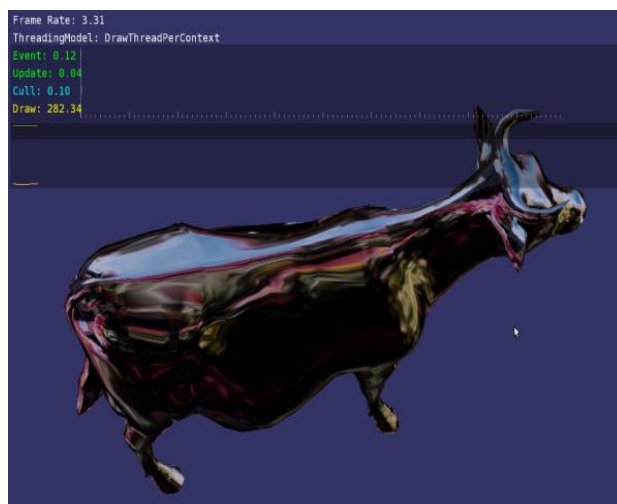


Figura3-7 Ventana de compilación del “Hola Mundo” con OSG

Fuente: Autores

3.6 Entorno de desarrollo integrado Eclipse

Es una herramienta de programación para el desarrollo de software de diversos tipos de código abierto, que es utilizado por varios usuarios, es apto para desarrollar interfaces de usuario por su sistema de ventanas, por lo que se lo ha seleccionado para realizar la programación del aplicativo 3D, para lo cual previamente se debe tener instalado y debidamente configurado Java JRE (Anexo Configuración de JRE) y Java JDK (Anexo Configuración de JDK). Para utilizar Eclipse Java y CPP (C++), tan solo se debe descargarlo de la página oficial de Eclipse[67], y descomprimirlo en cualquier carpeta o sitio que tenga permisos de ejecución(ver Anexo Instalación de Eclipse para Ubuntu y Instalación de Eclipse para CentOS 5).

Para el enlace dispositivo – entorno 3D se utilizará eclipse, sistema de código abierto en su versión para CPP (Plugin C++) en GNU/Linux, siendo necesario colocar las librerías (ver Librerías y paquetes de instalación C++) que requiera el programa[68].

Los componentes o plugins para eclipse son: Eclipse SDK, Eclipse CPP, Eclipse PlatformCore - controla el ciclo de vida de una aplicación de Eclipse, Standard WidgetToolkit (SWT), JFace.

Características

- Eclipse es una plataforma independiente. Debido a que Eclipse es en sí mismo escrito en Java, funciona en todas las principales plataformas, incluyendo Linux, Windows y Mac.
- Posee gran cantidad de plugins y un editor de código fuente con resaltado de sintaxis y con integración de herramientas externas.
- Apoya la gestión de proyectos para proyectos individuales y grupos de proyecto, así como las dependencias entre proyectos.
- Es un depurador de soporte para GNU con un gestor de proyecto integrado (*managed*) por lo que no hay que preocuparse por la creación (*makefile*).

A continuación se pueden apreciar algunas de las ventajas que proporciona este IDE así como sus desventajas.

Ventajas:

- Ofrece varias herramientas y configuraciones que permiten un mejor desarrollo de código y evita errores.
- Es más sencilla la asignación del *path* para las librerías a utilizar en el IDE de Java por lo que en la actualidad es un estándar en muchas industrias, siendo la más utilizada.

Desventajas:

- En el IDE para C++ no es sencilla la asignación del *path* de las librerías a utilizar para el desarrollo ya que deben ser colocadas de una en una tomando en cuenta mayúsculas y minúsculas.

3.7 3D Max Studio

3D Max Studio fue elegido porque permite importar y vincular archivos con extensión “.obj” los cuales serán incorporados en el aplicativo 3D para su desarrollo.

Programas 3ds Max no necesitan grandes presentaciones cuenta con más de cien herramientas de modelado, escultura y manipulación de objetos, y es compatible con la mayoría de formatos de la industria. Conocido durante años como 3DStudio, es uno de los mejores programas de modelado y renderizado 3D del mercado. Debido a la madurez y potencia de sus herramientas para la creación de gráficos de videojuegos, películas, imágenes infográficas, imágenes médicas y todo lo que se quiera crear para un mundo virtual.

Autodesk 3ds Max soporta los siguientes formatos:

3D Max Studio	
Formato	Descripción
3ds	3D Studio max: formato que respeta la asignación de materiales, texturas y la posición de la cámara, permite exportar ideas generadas en SketchUp con una fidelidad superior a la de los formatos diseñados para CAD
prj	Extensión de archivo de proyección: archivo que guarda la información referida al sistema de coordenadas
ai	Adobe Illustrator: Se basa en el formato PDF, cuando se guarda como .ai se puede editar al 100%, mantiene la transparencia y guarda una pre visualización al estilo PDF 1.4
dem	LandDEM: formato que pesa poco comparando con otros formatos de video (porque realmente no es un video)
ddf	LandDDF
dwg	AutoCAD Drawing: almacena la información de dibujo en tres dimensiones de forma vectorial.

3D Max Studio	
Formato	Descripción
dxf	AutoCAD <i>Drawing</i> almacena la información de dibujo en tres dimensiones
fbx	Kaydara: almacena el índice espacial de las entidades para los <i>shapefiles</i> que son inalterables (solo lectura).
ige	Erdas imagine: puede almacenar datos de banda única y multibanda tanto continuos como discretos
ls	Lightscape: hace simulaciones fotorrealísticas basadas en las condiciones reales de la luz
vw	Lightscape View
lp	LightscapePreparation: Long Play: formato para elegir mayor compresión o menor resolución para que entre en el disco duro
mtl	Wavefront Material
obj	Wavefront Object: formato de archivo 3D
shp	3D Studio Shape: archivo que almacena las entidades geométricas de los objetos
stl	STereoLithography: utiliza una malla de pequeños triángulos sobre las superficies para definir la forma del objeto
wrl	VRML: Inventor: son ficheros de texto que describen escenas en 3D en lenguaje VRM, pensado para ocupar poca memoria, admite animaciones, efectos de todo tipo, sonido y video
max	3D Studio: de tipo procedural, es decir, contiene instrucciones que deben seguir los diferentes plugins para construirlo.
chr	CHR File Extensión: puede tener distintos sonidos
drf	Viz Render

Tabla 3-3 Formatos de archivo soportados por 3D Max Studio

Fuente: Autores

Características:

- 3D Max 7 ofrece soporte para renderización con una amplia gama de funcionalidades que incluyen un normal mapping que es un acelerador de flujo de trabajo para juegos que agrega detalles extremos a los modelos; mental ray 3.3 para la integración de 3ds max con desempeño acelerado y mejor eficiencia de memoria; custom attributes colector es una interfaz que mejora la eficiencia cuando se animan múltiples atributos; edit poly modifie incrementa la rapidez y facilidad con las cuales se pueden crear, modificar y animar complejas superficies

poligonales; skin wrap deformer para el flujo de trabajo de animación de personajes, desempeño y escalabilidad interactivos para permitir la manipulación de alto desempeño de grandes cantidades de objetos; TurboSmooth es un algoritmo de suavizado para mayor desempeño de modelos de alta resolución,

- Posee nuevas herramientas de polígonos editables y una vista plana sombreada (plain shaded view) o de textura difusa, con un modo de navegación en primera persona a través de la escena.

A continuación se pueden apreciar algunas de las ventajas que proporciona este software de diseño así como sus desventajas.

Ventajas:

- Admite diferentes lenguajes de programación con más de un centenar de herramientas para el modelado y creación, permitiendo ambientar modelos 3D para hacerlos más realistas.
- Soporta varios plugins para adicionar efectos fácil y rápido, siendo algunos gratuitos por lo que es más sencillo de usar que otros programas de modelado y creación 3D.
- Permite aprovechar al máximo las herramientas y efectos disponibles para crear cualquier clase de objeto pudiéndolo ver en cualquiera de las vistas que el 3d maneja, así también se puede exportar a diferentes formatos según las necesidades.
- Posee una interfaz más intuitiva que otros modeladores 3D por lo que su manejo es más sencillo.

Desventajas:

- Necesita de un equipo potente para liberar todo el potencial que esta herramienta proporciona y para elevar la productividad.
- La curva de aprendizaje es complicada en un inicio siendo solventada con el trabajo continuo
- El software es privativo pero posee una evaluación de prueba (shareware).

Se utiliza el emulador “*Wine*” el cual permite que muchas aplicaciones para Windows sean ejecutadas sin modificaciones en sistemas operativos como Linux, para que funcione 3D Max Studio. Se procede a efectuar la instalación en Ubuntu 11.04, se ha escogido esta versión de 3D Max Studio por ser de fácil instalación y configuración (Anexo Instalación de 3D Max Studio 7).

En la Figura 3-8 se puede apreciar la interfaz gráfica de este diseñador:

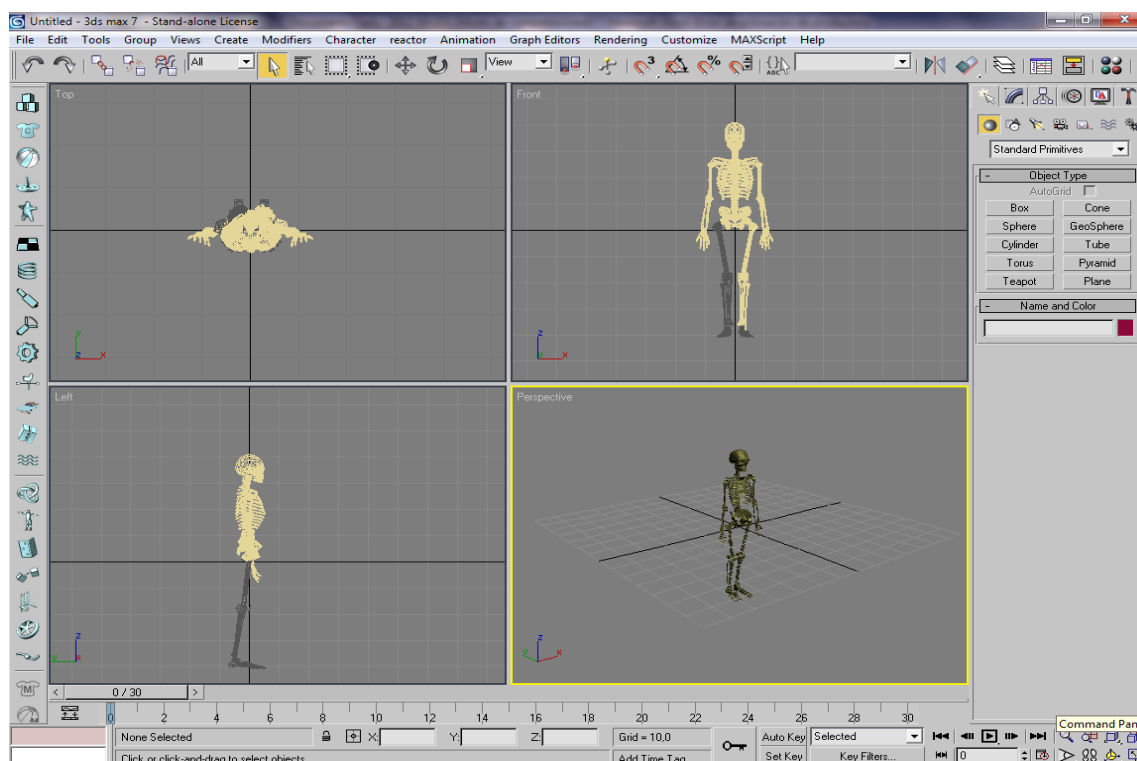


Figura 3-8 Interfaz gráfica de 3D Max Studio

Fuente: Autores

3.8 Adobe Photoshop

La aplicación Adobe Photoshop permite manipular y crear imágenes de calidad fotográfica de una forma profesional, se la eligió para la edición de las imágenes que recubrirán y formarán parte del cuerpo humano.

A la hora de trabajar con imágenes se pueden distinguir dos tipos de imágenes. Las imágenes vectoriales se componen de líneas y curvas definidas

matemáticamente y las imágenes de mapa de bits están compuestas por una rejilla o mapa de pequeños cuadrados denominados píxeles.

Los programas de dibujo generan imágenes de tipo vectorial. A veces se llaman "programas de dibujo orientado a objeto", ya que las imágenes comprenden objetos independientes, líneas definidas matemáticamente y figuras.

Es la herramienta que dará tratamiento a las imágenes como: piel, músculos, esqueleto, sistema respiratorio, que serán acopladas y envolverán a los objetos en 3D Max Studio.

Formatos de archivo

Photoshop soporta muchos tipos de archivos de imágenes además de tener formatos de imagen propios, los formatos soportados por Photoshop [69] son:

Adobe Photoshop	
Formato	Descripción
PSD, PDD	Formato estándar de Photoshop con soporte de capas.
PostScript	No es exactamente un formato, sino un lenguaje de descripción de páginas. Utiliza primitivas de dibujo para poder editarlo.
EPS	Se utiliza para situar imágenes en un documento, compatible con programas vectoriales y de autoedición.
DCS	Permite almacenar tipografía, tramas. Utiliza para filmación en autoedición.
Prev. EPS TIFF	Permite visualizar archivos EPS que no se abren en Photoshop, por ejemplo los de <i>quarkxpress</i> .
BMP	Formato estándar de Windows.
GIF	Permite almacenar un canal alfa para dotarlo de transparencia, y salvarlo como entrelazado para que al cargarlo en la Web lo haga en varios pasos.
JPEG	Es un factor de compresión muy alto y buena calidad de imagen.
TIFF	Una solución creada para pasar de PC a MAC y viceversa.
PICT	Desde plataformas MAC se exporta a programas de autoedición como QuarkXPress.
PNG	Formato con mayor calidad. Soporta transparencia y colores a 24 bits.
PDF	Formato original de Acrobat. Permite almacenar imágenes vectoriales

Adobe Photoshop	
Formato	Descripción
	y mapa de bits.
IFF	Se utiliza para intercambio de datos con amiga.
PCX	Formato solo para PC. Permite colores a 1, 4, 8 y 24 pixeles.
RAW	Formato estándar para cualquier plataforma o programa gráfico.
TGA	Compatible con equipos con tarjeta gráfica de <i>truevision</i> .
Scitex CT	Formato utilizado para documentos de calidad profesional.
Filmstrip	Se utiliza para hacer animaciones. También se puede importar o exportar a <i>premiere</i> .
FlashPix	Formato originario de <i>kodak</i> para abrir de forma rápida imágenes de calidad superior.

Tabla 3-4 Formatos de archivo soportado en Photoshop

Fuente: Autores

A continuación se pueden apreciar algunas de las ventajas que proporciona este software de diseño así como sus desventajas.

Ventajas:

- Posee un soporte de imágenes de alto rango dinámico (HDR). La vista de diseño de Photoshop es renderizada por la tarjeta de video y acelerada por hardware, además posee una interfaz clásica permitiendo un fácil manejo.
- Elimina elementos de una imagen automáticamente y rellena el fondo, también se puede rotar el *canvas* sin modificar la imagen.
- Las capas de ajuste hacen que el proceso sea reversible y no destructivo por lo que se puede regresar a la imagen original.
- Las herramientas de *dodge* y *burn* protegen los tonos permitiendo mover los elementos de la imagen que sea requerida sin modificar otros elementos siendo muy potente incluso para uso profesional.
- Es posible achicar o agrandar una imagen y en vez de que ésta se deforme se redimensiona de manera inteligente manteniendo el contenido importante en la misma proporción.

- Posee atajos de teclado para un fácil manejo de la herramienta, además para un mejor entendimiento se pueden encontrar tutoriales en video y contenido online.

Desventajas:

- Algo difícil de utilizar al principio hasta familiarizarse con el entorno
- Posee un alto consumo de recursos debido a su gran potencia, requiere una PC con una buena capacidad de procesamiento y una tarjeta de video.
- Software privativo pero posee una evaluación de prueba (shareware)

Se utiliza Adobe Photoshop en su versión de prueba mediante “*Wine*” para la edición de texturas que vayan acorde a los tejidos del cuerpo humano.

3.9 Clúster Rocks

Rocks se basó inicialmente en la distribución Red Hat Linux, sin embargo las versiones más modernas de Rocks están basadas en CentOS, con un instalador Anaconda modificado, que simplifica la instalación 'en masa' de computadoras, por esta razón se lo utilizará para distribuir y paralelizar el aplicativo.

A continuación se describe el sistema de un Clúster Rocks

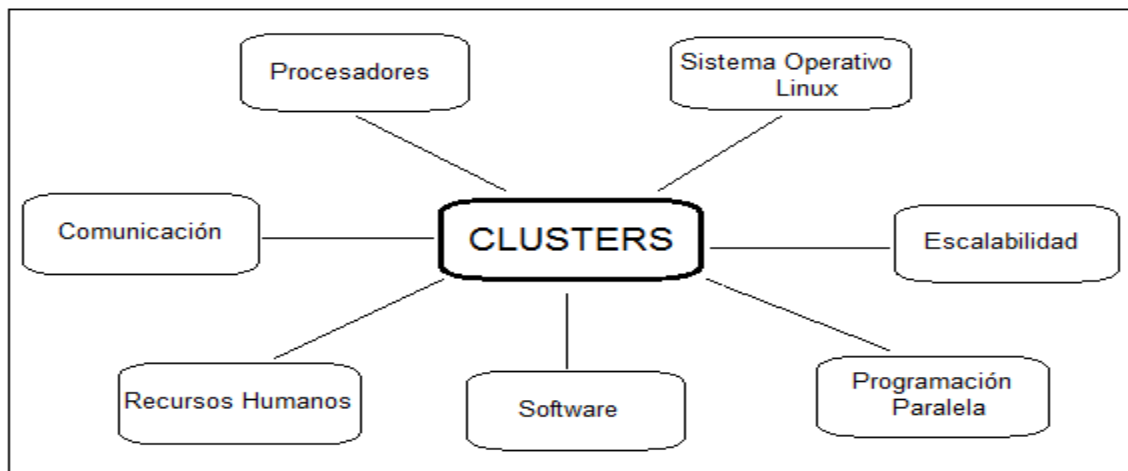


Figura 3-9 Sistema de un Clúster

Fuente: Autores

Elementos de un Clúster:

- **Procesadores:** Los procesadores de máquinas a las cuales se puede acceder proporcionan un rendimiento similar a los procesadores de una supercomputadora, donde cada procesador posee gran cantidad de cache, así como de altas velocidades y bajo costo.
- **Comunicaciones:** Para la interconexión entre los equipos se puede utilizar redes como Ethernet, Myrinet, Gigabit, con lo cual se consigue un mayor ancho de banda disponible para la comunicación con bajas latencias.
- **Sistemas Operativos:** Puede utilizarse en cualquier sistema operativo como Linux ya que posee gran estabilidad conjuntamente asociado a un alto rendimiento en cuanto a manejo de memoria, así como de I/O eficiente posibilitando el realizar ajustes mínimos a los parámetros de los dispositivos para mejorar su rendimiento.
- **Software:** A partir de la aparición de los procesadores con *HiperThreading* (HT), la programación y la difusión de software se ha desarrollado exponencialmente, teniendo mayor cantidad de medios para las diferentes disciplinas científicas.
- **Recursos Humanos:** El elemento humano capacitado en la administración y manejo de recursos proporciona un entorno amigable para los usuarios que pretendan utilizar el clúster.

Características:

- Mejora de seguridad ya que solo con autorización expresa el Applet puede ser leído desde disco, porque requiere de la llamada al URL con lo que no pueden descargarse las imágenes a menos que se le otorgue los permisos necesarios para ello.
- Posee alta disponibilidad, escalabilidad, altas prestaciones a bajo costo y un mejor diseño. Sistema de arranque adecuado al procesar los datos del Applet.

A continuación se pueden apreciar algunas de las ventajas que proporciona este clúster así como sus desventajas.

Ventajas:

- Incluye herramientas llamadas *rolls* y permite un acceso transparente a los nodos o auto montado de directorios compartidos.
- Automatización de procesos en el manejo de un Clúster.
- Proporciona el sistema operativo en el cual trabaja, siendo fácil y manejable, incluye instrucciones útiles para la administración.

Desventajas:

- Los requisitos mínimos necesarios de hardware pueden variar según la versión y herramientas a utilizar.
- La velocidad de la red será un limitante en la velocidad del Clúster por ejemplo una Fast Ethernet con 100 Mbps como ancho de banda.

Programación Paralela y Distribuida

- La ejecución paralela implica un número de tareas concurrentes ejecutándose simultáneamente de forma síncrona o asíncrona en base a una topología de conexión siguiendo un modelo maestro²⁶/esclavo²⁷ o distribuido con unas tasas de comunicación y un intercambio de mensajería.
- Permite un fino control de la comunicación la cual se hace explícitamente mediante mensajes que contienen información, por lo que la programación resulta más compleja.

Modelo Maestro-Esclavo[70]

El paradigma maestro-esclavo tiene un *thread*/ proceso llamado maestro que pone en marcha otros procesos llamados esclavos a los cuales les asigna trabajo y consigue las soluciones parciales que se generan.

Modelo MPI: Permite programar y coordinar tareas que se ejecutan en paralelo en diferentes procesadores (nodos), los cuales se comunican mediante mensajes. El

²⁶Maestro: Distribuye el trabajo entre los trabajadores

²⁷Esclavo: Trabajador (mismo código pero con diferentes datos)

protocolo MPI esta implementado en lenguaje C (MPICH, openMPI) y el Roll HPC de Rocks incluye openMPI.[71]

La arquitectura de Open MPI posee una estructura modular formada por *frameworks* y componentes, existe un *daemon* en cada nodo, para crear el entorno paralelo y posee soporte de tolerancia a fallos.

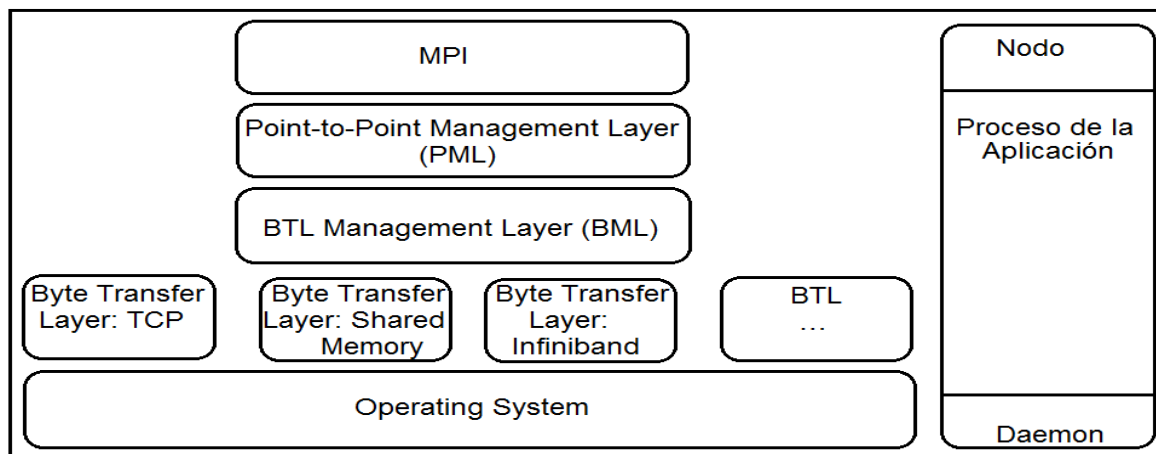


Figura 3-10 Arquitectura OpenMPI

Fuente: Fialho Leonardo, Incorporando radica OpenMPI, 2012[72]

Ventajas:

- Open MPI es un proyecto de código abierto de implementación MPI-2 que es desarrollado y mantenido por un consorcio de investigación académica y socios de la industria, combina conocimientos, tecnologías y recursos de toda la comunidad informática de alto rendimiento con el fin de construir la mejor librería MPI disponible.
- Es una ventaja para los vendedores de sistemas y software, desarrolladores de aplicaciones e investigadores de ciencias de la computación, no es necesario modificar el código de la aplicación para realizar la asignación manual en OpenMPI.

Desventaja:

- La asignación es estática y no puede modificarse durante la ejecución.

Una vez considerados los requisitos se procede a la instalación del FrontEnd para lo cual se tendrá que revisar las conexiones de red externas e internas puesto que Rocks asume que la interfaz eth1 será aquella que está conectada a la red externa y la eth0 a la red privada del Clúster. Luego se procederá a la instalación de los nodos en el Clúster, logrando que el FrontEnd detecte las peticiones de instalación, registre la MAC de los mismos y les asigne un nombre.

3.10 PROCESO DE DESARROLLO

3.10.1 Roles definidos en la metodología XP

Los desarrolladores del aplicativo son dos personas por lo que los roles definidos en la metodología XP fueron asignados a ellos. En algunos casos se ha solicitado el apoyo de otros desarrolladores que harán las veces de:

- **Programador:** Producir el código del sistema y colocar la textura al resto de imágenes del cuerpo humano.
- **Cliente:** Asignar la prioridad a las historias de usuario y decidir cuáles serán las que se implementarán en cada iteración.
- **Encargado de pruebas:** Ejecutar las pruebas regularmente informando los resultados y apreciaciones al equipo de desarrollo.
- **Encargado de seguimiento:** Suministrar la realimentación al equipo y realizar el seguimiento en el avance de cada iteración.
- **Entrenador:** Proveer guías al equipo de manera que se apliquen las prácticas XP, siguiendo el proceso correctamente.

3.10.2 Planificación

A. Historias de Usuarios

El cliente establece la prioridad de cada historia de usuario y las características que el sistema deberá poseer para el proyecto “Sistema de Entrenamiento Virtual para

Medicina”, y conjuntamente con el programador se establecerá el contenido a desarrollar.

Para detallar las historias de usuario se ha utilizado los siguientes elementos:

- **Código y Número:** Las historias se reconocerán por el código HU seguido del número de historia.
- **Usuario:** Quien va a utilizar el resultado de la implementación del requerimiento.
- **Nombre de la Historia:** Nombre con el cual se diferenciará de las otras historias de usuario.
- **Prioridad en Proyecto:** Se indica la prioridad de la implementación con: alta, media, baja.
- **Iteración:** Representa el número de iteración por cada requerimiento que sea modificado o agregado.
- **Programador Responsable:** Nombre de los encargados para la implementación del requerimiento.
- **Descripción:** Se describe a breves rasgos el requerimiento pedido por el cliente.
- **Observaciones:** Si existen se las debe escribir para que el desarrollador la tome en consideración.

En la tabla siguiente se indica la historia de usuario para la creación y edición de las imágenes en 3D:

Historia de Usuario	
Código y Número: HU1	Usuario: Cliente
Nombre de la Historia: Creación y edición de imágenes en 3D	
Prioridad en Proyecto: Alta	Iteración: 2
Programador Responsable: Inaquiza Blanca, Jácome Mauricio	
Descripción: Diseño e implementación de las imágenes con extensión .obj	
Observaciones:	

Tabla 3-5 Historia de Usuario HU1

Fuente: Autores

En la tabla siguiente se indica la historia de usuario referente a la Creación del aplicativo 3D:

Historia de Usuario	
Código y Número: HU2	Usuario: Cliente
Nombre de la Historia: Creación de un aplicativo para manipular imágenes 3D (.obj)	
Prioridad en Proyecto: Alta	Iteración: 2
Programador Responsable: Inaquiza Blanca, Jácome Mauricio	
Descripción: Diseño e implementación del aplicativo 3D.	
Observaciones:	

Tabla 3-6 Historia de Usuario HU2

Fuente: Autores

En la tabla siguiente se indica la historia de usuario para el desarrollo de la página Web:

Historia de Usuario	
Código y Número: HU3	Usuario: Cliente
Nombre de la Historia: Creación de una página Web para manipulación del Applet desarrollado	
Prioridad en Proyecto: Media	Iteración: 1
Programador Responsable: Inaquiza Blanca, Jácome Mauricio	
Descripción: Diseño e implementación de una página Web que contenga un diseño agradable al usuario, siendo amigable y de fácil uso.	
Observaciones: Las texturas incorporadas a las imágenes 3D requieren memoria en procesador.	

Tabla 3-7 Historia de Usuario HU3

Fuente: Autores

En la tabla siguiente se indica la historia de usuario para la iteración dispositivo – entorno 3D:

Historia de Usuario	
Código y Número: HU4	Usuario: Cliente
Nombre de la Historia: Creación del software que gestione los datos provenientes del dispositivo electrónico y envíe dicha información a los aplicativos 3D.	
Prioridad en Proyecto: Alta	Iteración: 1
Programador Responsable: Inaquiza Blanca, Jácome Mauricio	
Descripción: Diseño e implementación de un aplicativo que permita interactuar entre el dispositivo electrónico y los aplicativos para manipular las imágenes 3D.	
Observaciones: Previamente debe realizarse la instalación de las librerías para manejo del puerto USB en el entorno a ejecutar, con los respectivos permisos de ejecución para su funcionamiento.	

Tabla 3-8 Historia de Usuario HU4

Fuente: Autores

En la tabla siguiente se indica la historia de usuario para incluir el aplicativo en el Clúster.

Historia de Usuario	
Código y Número: HU5	Usuario: Cliente
Nombre de la Historia: Aplicativo desarrollados incluido en el Clúster	
Prioridad en Proyecto: Alta	Iteración: 2
Programador Responsable: Inaquiza Blanca, Jácome Mauricio	
Descripción: Configuración de herramientas para el uso y paralelización de la aplicación 3D	
Observaciones: Las imágenes 3D para ser procesadas por el Clúster requieren tarjeta gráfica	

Tabla 3-9 Historia de Usuario HU5

Fuente: Autores

B. Plan de Historias de Usuarios

Se elabora tomando en cuenta el orden de implementación de cada historia de usuario y de las iteraciones con las que cuente.

Tiempo estimado				
Código y Número	Historia de Usuario	Semanas Estimadas	Días Estimados	Horas Estimadas
HU1	Creación y edición de imágenes en 3D	8	40	5
HU2	Creación de un aplicativo para manipular las imágenes 3D (.obj)	8	40	5
HU3	Creación del software que enlace los datos provenientes del dispositivo electrónico y envíe dicha información a los aplicativos 3D.	8	40	5
HU4	Creación de una página Web para manipulación del Applet desarrollado	8	40	5
HU5	Los aplicativos desarrollados deben estar incluidos en el Clúster	24	180	8
Tiempo Estimado Total		56	340	28

Tabla 3-10 Estimación de Historia de Usuario.

Fuente: Autores

C. Iteraciones

Cada iteración por desarrollar estará basada en las Historias de Usuario de acuerdo a los procesos a ser realizados.

HU1: Creación y Edición de Imágenes en 3D

De acuerdo a la Tabla 3-5 Historia de Usuario HU1 previamente revisada se establecerá la estructura del sistema a ser utilizado durante el resto del proyecto para la creación y manipulación de las imágenes 3D.

Iteración 1 – Modelado y Diseño en 3D Max Studio

3D Max Studio es un programa que permite la creación de animaciones, logotipos, diseños 3D, videojuegos, publicidad, efectos especiales y en menor medida la arquitectura.

Para efectuar la creación y edición de imágenes en 3D de las partes del cuerpo humano se tendrán que conocer las herramientas que 3D Max Studio posee, con esto se optimizará tiempo y recursos al proceder con la manipulación y modelado del objeto 3D, familiarizándose con el entorno o pantalla de inicio del programa en el que se trabajará.

Tener en cuenta las utilidades que proporcionan cada una de las herramientas y estructuras 3D para la manipulación del objeto 3D (ver Anexo Herramientas de 3D Max Studio 7).

El diseño e implementación de las imágenes, con extensión .obj, es la primera tarea en el desarrollo del aplicativo ya que es la base fundamental para el funcionamiento y éxito del software. Para el desarrollo se requiere de lo siguiente:

- a) **Clasificación de imágenes 3D de las partes del cuerpo humano:** Para proceder con la elaboración del aplicativo y con el fin de optimizar tiempo y recursos en la creación de imágenes 3D, se realizará una búsqueda de imágenes del cuerpo humano en Internet para posteriormente seleccionarlas y clasificarlas ajustando las mejores al aplicativo.

Las imágenes obtenidas se ajustarán unas con otras para integrarlas entre sí, para editarlas, es decir, acoplarlas y ajustarlas a los requerimientos definidos.

Antes de importar archivos en 3D Max es necesario tener en cuenta algunas consideraciones:

Evitar que la geometría esté muy alejada del punto de origen de las coordenadas.

Evitar tener geometrías demasiado alejadas unas con respecto a otras.

Importar sólo la geometría necesaria, sin cotas, textos, ejes, niveles.

Organizar todo en capas como por ejemplo: piel, músculos, sistema respiratorio, órganos.


- b) **Estructuración de las Imágenes:** Cada imagen debe ir ubicada en su posición correcta, empezando desde los huesos que serán la base para estructurar y colocar cada pieza en su lugar hasta situar el elemento final que es la piel.
- c) **Modelado y fusión de las partes del cuerpo humano:** Debido a que el programa ofrece algunas estructuras 3D ya creadas solo se debe colocar en el visor y escoger el tamaño, siendo de utilidad al momento de crear polígonos ya que se creará la forma para luego estirarlos y obtener el volumen deseado, con lo cual se eligieron las geometrías acorde al elemento a desarrollar como:

- “*Standard Primitives*” (Primitivas estándar): Son formas básicas de las cuales se utilizará “*Cylinder*” (cilindro), “*Sphere*” (esfera), *Tube* (tubo).
- “*Extended Primitives*” (Primitivas Extendida): Son formas complejas de las cuales se utilizará “*Chamferbox*” y “*Capsule*”.
- “*DynamicsObjects*” (Objetos Dinámicos): El objeto de resorte es un objeto dinámico en forma de un muelle en espiral del cual se utilizará “*Spring*”.

Cada una de estas formas servirá para la creación de varios órganos y conductos en 3D que se añadirán al cuerpo humano posteriormente.

Para que las imágenes en 3D puedan ser acopladas al resto de imágenes, se procede a realizar algunos ajustes:

Importar el archivo de la ubicación en la que se encuentra, para ello dirigirse a “File/ -Import/ -‘nombre_archivo’.obj/ - Abrir”.

- Si algún archivo .obj que se obtuvo posee iluminación (*lights*), se tendrá que eliminar, para ello se seleccionará al objeto “*lights*” y se procederá a suprimirlo ya que el aplicativo a desarrollar no requiere de su luz.
- Se importará el archivo .obj que será el *pivot* central, es decir, el esqueleto, el cual servirá de base para el resto de imágenes 3D a editar y diseñar, de esta manera se acoplará cada imagen.
- Lo primero a editar será el tamaño de la imagen ver Figura 3-11, para lo cual se realizará un escalado, luego aplicar rotación y traslación usando las herramientas dadas para esta edición. 
- Se procederá a incorporar la imagen dentro del esqueleto para su posterior edición y modificación.

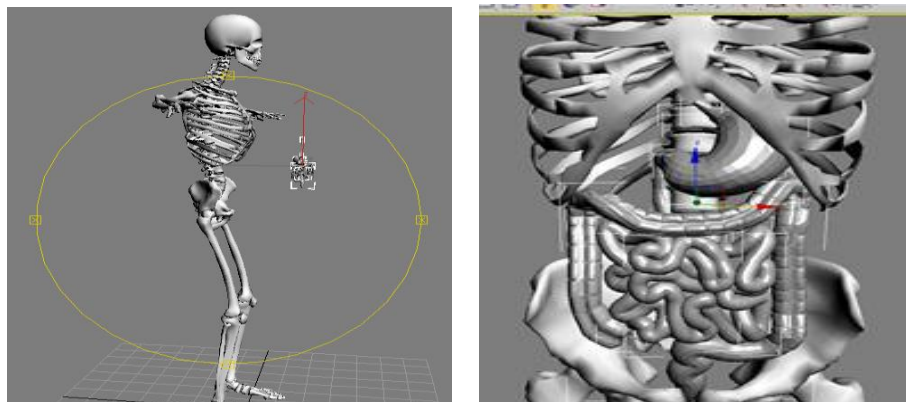


Figura 3-11 Ventanas para acoplar y editar las imágenes 3D

Fuente: Autores

En esta iteración se obtiene la aceptación del cliente sobre los avances presentados por lo que se procede al desarrollo de la aplicación.

Diseño

Para el diseño de las imágenes 3D se sigue lo que plantea la metodología ágil XP, evitando soluciones complejas y se trabajando en una iteración a la vez.

Iteración 1. Diseño e implementación de las imágenes con extensión .obj

A. Rediseño de imágenes 3D

Las imágenes 3D tendrán que ser convertidas a puntos de vértice (ver Figura 3-12) cada una de ellas para obtener los “Vertex” y ajustarlos a conveniencia de tal forma que se acoplen sin problema alguno al pivot central “esqueleto” ya que suelen no encajar con este.

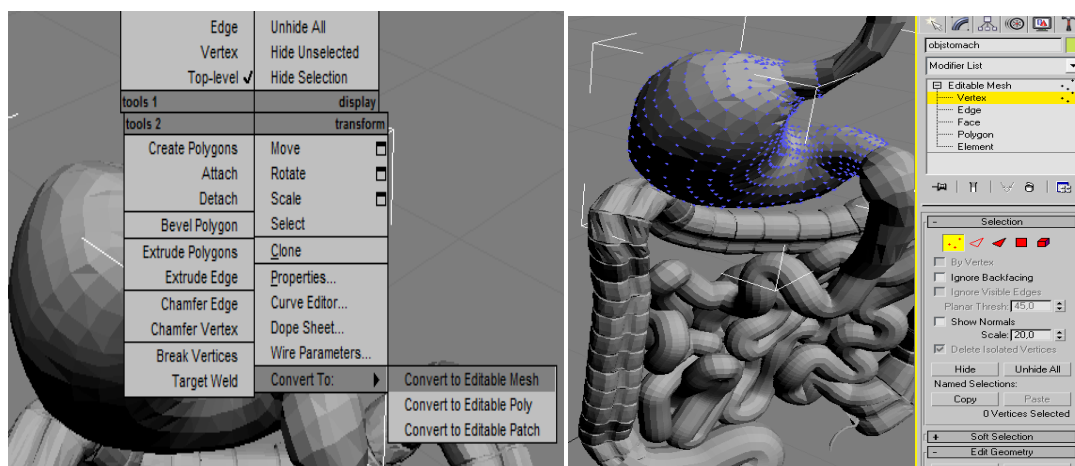



Figura 3-12 Ventana para convertir a puntos de vértice

Fuente: Autores

Con los puntos de vértice obtenidos se manipulará ya sea rotando, escalando y trasladando  a la ubicación acorde a cada órgano.

B. Creación de imágenes 3D

Para la creación de las imágenes en 3D Max se crea un objeto seleccionando la geometría basado en el Atlas de Anatomía Humana[73]; los puntos de vértice que poseen estas imágenes facilitará el manejo de la misma al momento de ajustar cada punto y formar el órgano diseñado, por ejemplo: para elaborar el órgano “bazo” se eligió la geometría “Cylinder” ya que este posee varios puntos de vértice a los cuales se los puede ajustar hasta obtener el diseño adecuado, el tipo “Tube” para el órgano “pene” por tener una forma similar (ver Figura 3-13) y para ello se seleccionó “*Stándard Primitives*”:

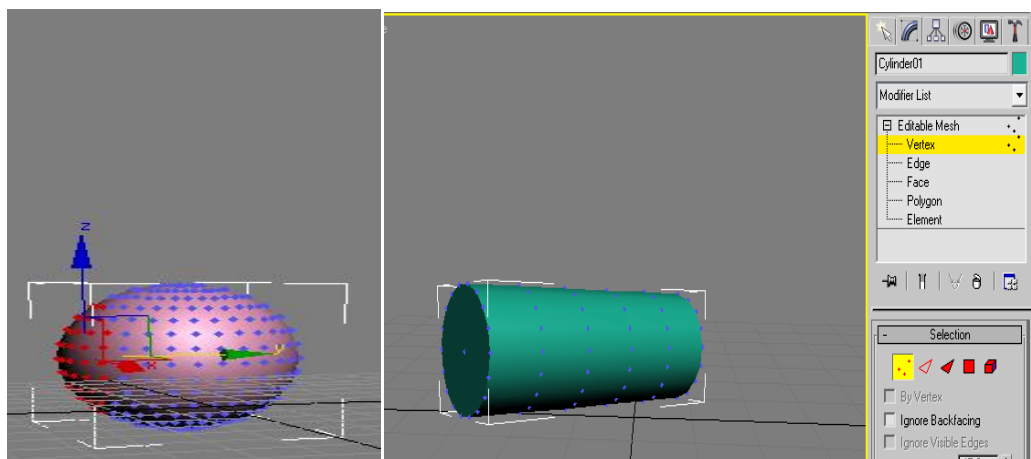


Figura 3-13 Ventana de selección de geometrías Standard Primitives

Fuente: Autores

Se eligieron otras geometrías con menos complejidad en la creación de otros órganos cuyas formas son tipo capsulares como “*ChamferBox*” para órganos, por ejemplo: la próstata de tipo “*Capsule*” para la vejiga, los testículos (ver Figura 3-14), para lo cual se seleccionó “*Extended Primitives*”:

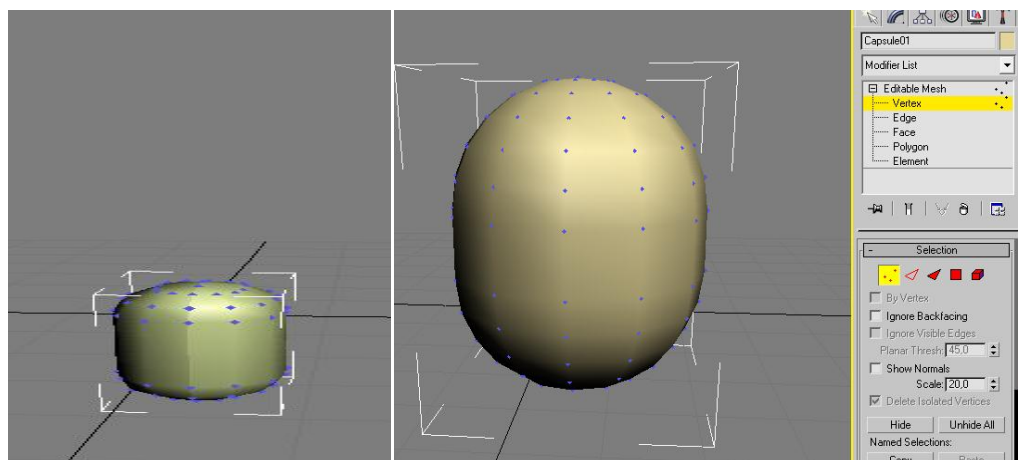


Figura 3-14 Ventana de selección de geometría Extended Primitives

Fuente: Autores

Para la elaboración de otras partes del cuerpo que requieren formas alargadas como las conexiones de los conductos espermáticos y de la bilis se ha elegido “*DynamigsObjects*” tipo “*Spring*” por ser el que mejor se acomoda a la estructura de dichos conductos(ver Figura 3-15).

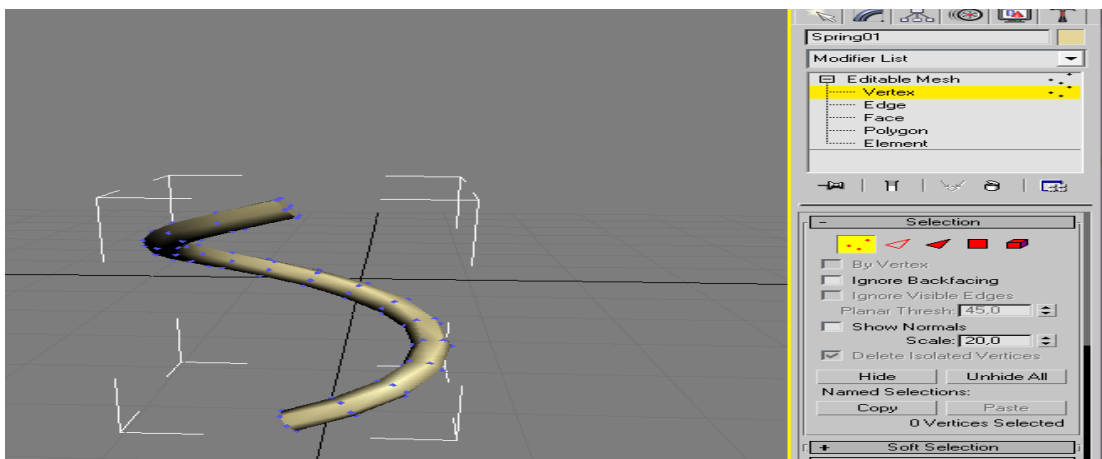


Figura 3-15 Ventana de selección de geometría "Spring"

Fuente: Autores

C. Textura para diseño 3D

Para que los objetos tengan una textura y se asemeje de mejor manera a un órgano real se buscará en Internet imágenes con estructuras similares para posteriormente editarlas. En 3D Max Studio se importará el elemento 3D con extensión .obj, luego se seleccionará todo el elemento .obj y en la pestaña de modificaciones se seleccionará y añadirá "Unwrap UVW" con lo que se podrá renderizar para luego guardarla y exportarla; con la herramienta Adobe Photoshop se abrirá la imagen .png para editarla, añadiendo las texturas y volviendo a guardar el archivo pero con extensión .jpg, para mayor detalle ver Anexo Añadir Texturas a objetos 3D.

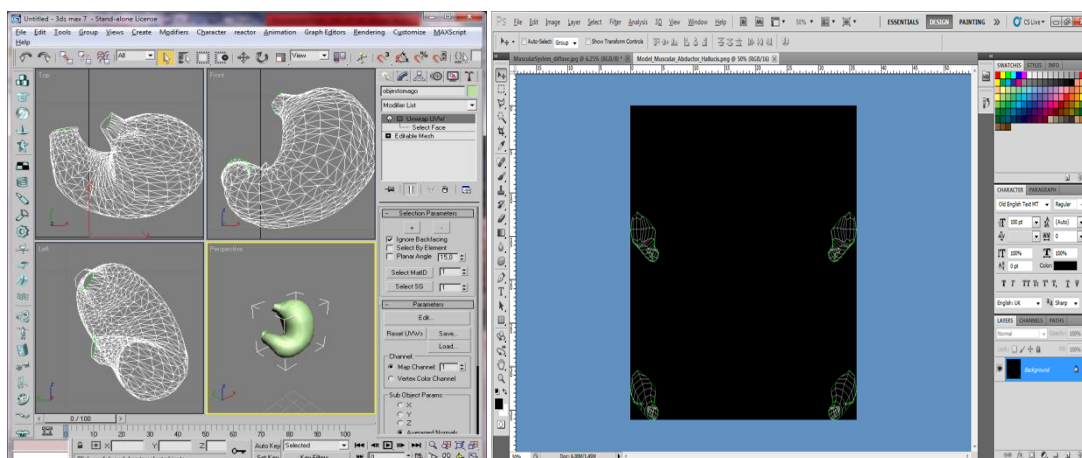


Figura 3-16 Ventana de modificaciones con UNWRAP UVW y textura en Photoshop

Fuente: Autores

D. Fusión Diseño 3D con texturas

Para colocar las texturas se importarán las imágenes con 3D Max Studio. Abrir el editor de materiales pulsando la tecla “M” que es el acceso rápido donde se desplegará la ventana del editor de material procediendo a seleccionar “*Bitmap*” para elegir la imagen a colocar como textura, abrirla con lo que se podrá observar que el material ha cubierto el primer recuadro del editor de materiales, para luego colocar la textura en la imagen .obj, como muestra la Figura 3-17 para mayor información ver Anexo Fusión Diseño 3D-Texturas.

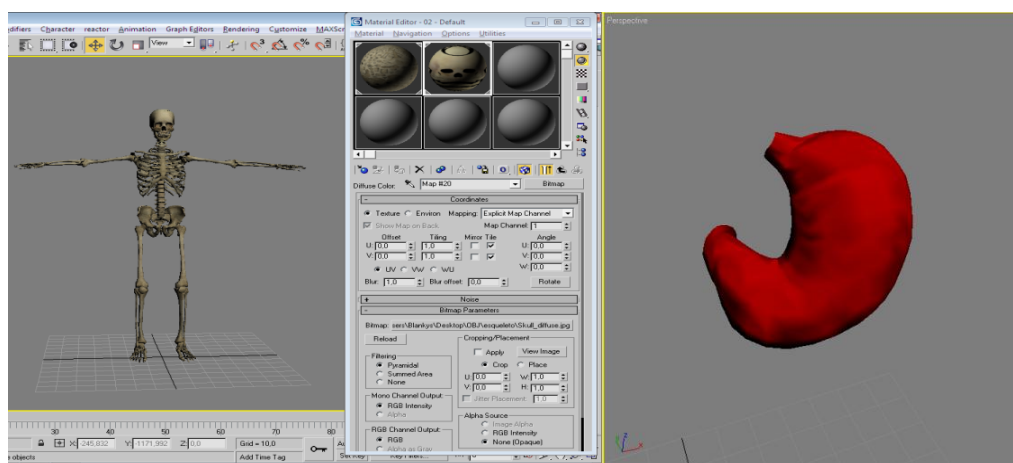


Figura 3-17 Ventana de textura asignado a la imagen.

Fuente: Autores

Este diseño fue presentado al cliente quien aprueba el trabajo por lo que se procede a su elaboración.

Codificación

Iteración 1. Desarrollo e implementación de las imágenes con extensión .obj

Para la edición de las imágenes 3D se procede a manipular los puntos de vértice que sean necesarios y se irán ajustando al pivot central “*esqueleto*” de tal manera que cada objeto se incorpore a la estructura(ver Figura 3-18), dependiendo del tipo de órgano a editar será necesario ir acoplando partes del cuerpo humano como: sistema respiratorio, músculos, órganos, piel y se operarán los extremos de los órganos con

cuidado ya que como son interconectados; al transformar un órgano se desarticula el siguiente órgano conectado, por lo que hay que tener en cuenta los puntos extremos.

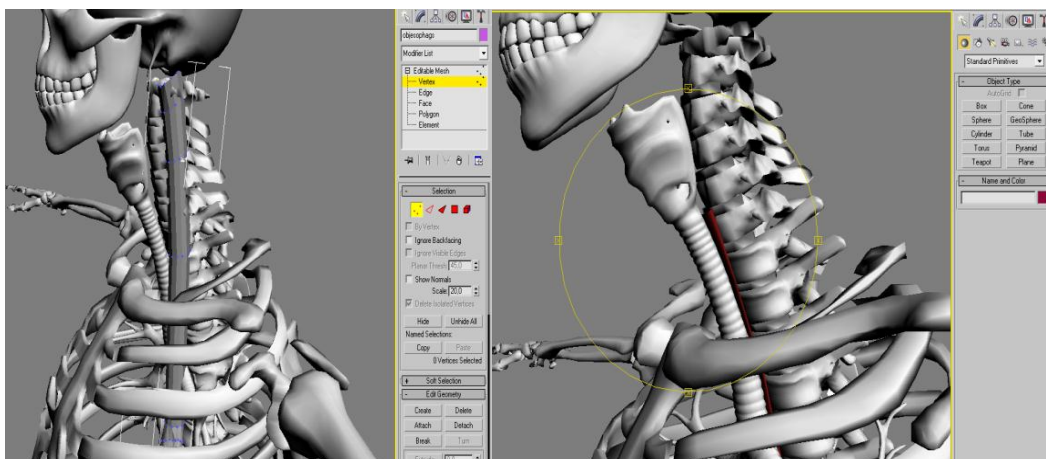


Figura 3-18 Ventana de acoplado de imagen editada

Fuente: Autores

Para la creación de las imágenes se empezó tomando los puntos de vértice y dando forma al objeto, cada “*vertex*” fue seleccionado en conjunto o individual hasta que la geometría se asemeje al órgano deseado, por ejemplo para la elaboración del órgano “bazo” se tomaron en conjunto los “*vertex*” y se lo fue manipulando dándole forma de media luna, luego se redujo la parte exterior y se dio una forma hueca en la parte interior para mayor detalle y realismo.

Terminado el detalle se tiene el objeto listo (ver Figura 3-19) para adjuntarlo al resto de órganos y acoplarlo al cuerpo humano para su posterior edición que será requerido, procediendo de igual forma a lo detallado anteriormente.

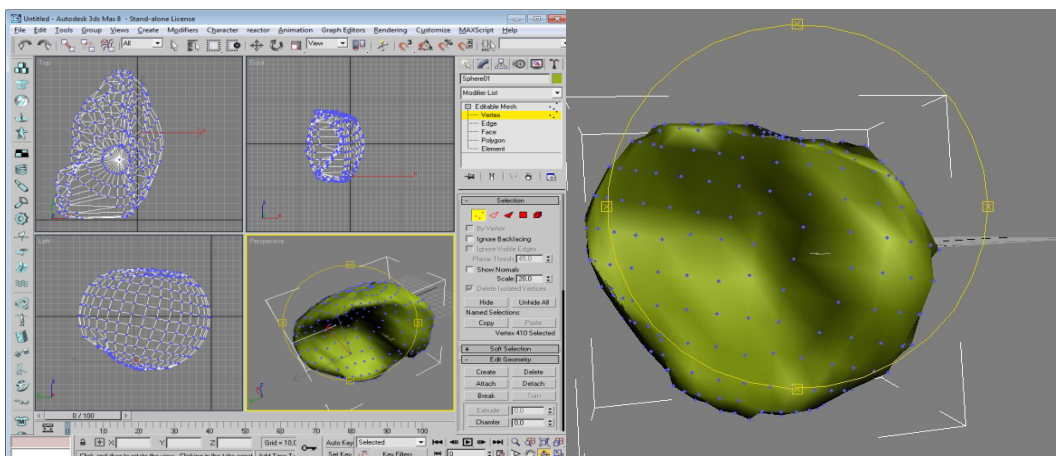


Figura 3-19 Ventana de órgano detallando y finalizando su construcción

Fuente: Autores

Para la elaboración del resto de órganos se procederá de la misma manera que en la elaboración del órgano “bazo”.

Pruebad e Unidad

Las imágenes de alta calidad como la Figura 3-20 contienen claridad (bump.jpg), profundidad (specular.jpg) y realismo (diffuse.jpg) para las diferentes texturas, las cuales están guardadas en un único archivo .JPG permitiendo envolver a cada uno de los objetos 3D.

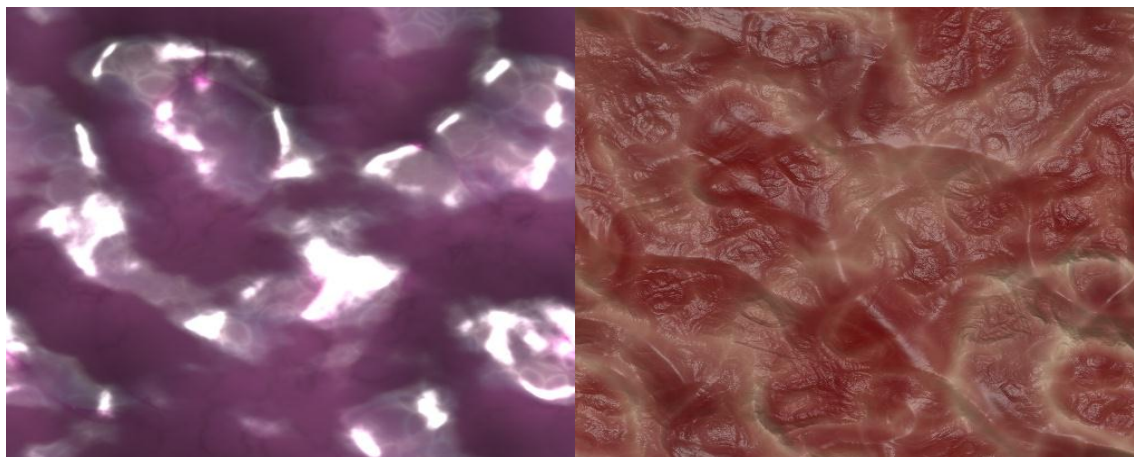


Figura 3-20 Ventana con textura de órganos

Fuente: Autores

Las imágenes enviadas desde 3D Max Studio fueron editadas, cortadas y colocadas una por una en archivos diferentes con los nombres respectivos como muestra la Figura 3-21:

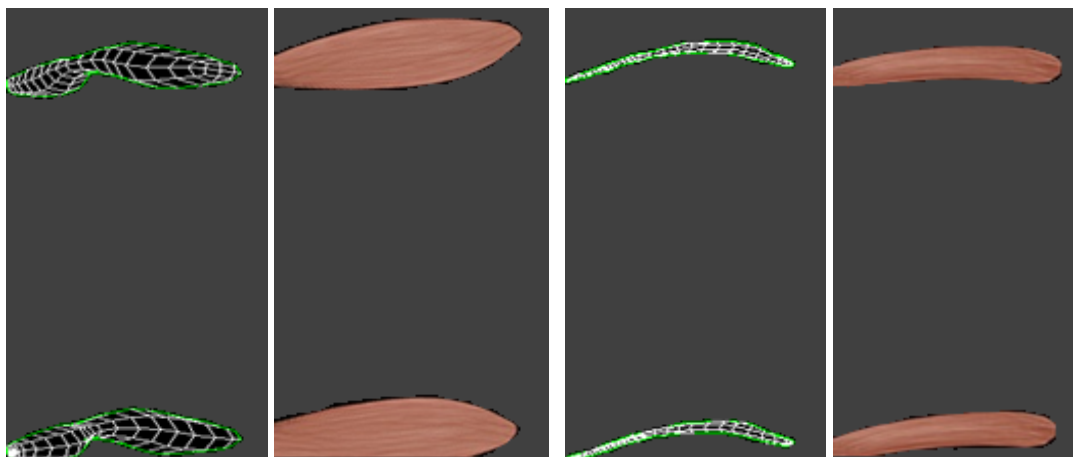


Figura 3-21 Ventana con texturas editadas

Fuente: Autores

Una vez que fueron creadas y editadas todas las imágenes de las partes del cuerpo humano son acopladas al resto de objetos 3D(esqueleto, órganos, sistema respiratorio, músculos y piel), dando como resultado la Figura 3-22.

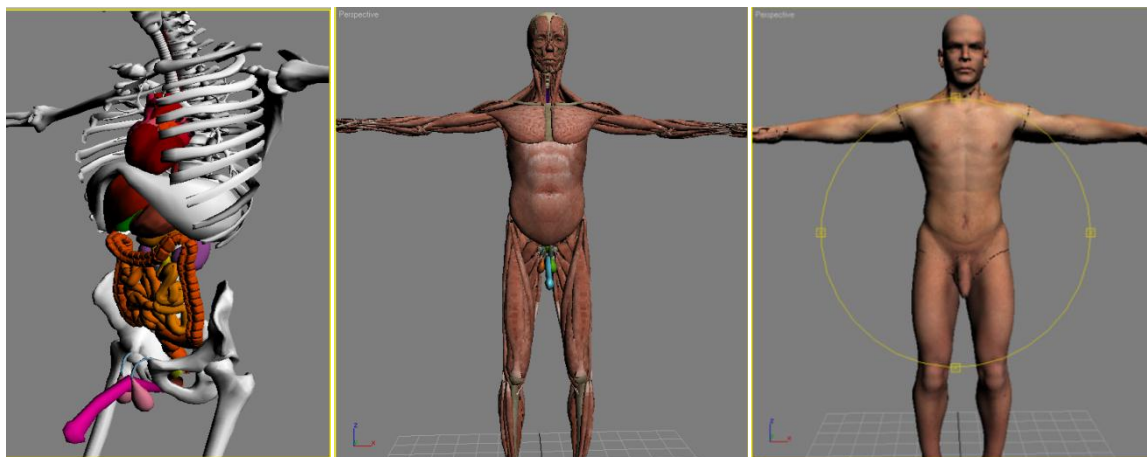


Figura 3-22 Ventana de órganos final junto al esqueleto

Fuente: Autores

Comprobación de errores

Error 01: Las imágenes de las texturas tienen un tamaño muy grande.

Se requiere manipular imágenes que contienen: claridad (bump.jpg), profundidad (specular.jpg) y dan realismo a los objetos 3D (diffuse.jpg), el tamaño de estas texturas en megabytes y en pixeles es de 8192 x 8192, siendo muy grande para el Applet, ya que no tiene capacidad para desplegar imágenes de este tamaño. El error que java reporta es Java *heap space* como muestra la Figura 3-23:

```
java.lang.OutOfMemoryError: Java heap space
    at javax.media.j3d.ImageComponentRetained$ImageData.<init>(ImageComponentRetained.java:2233)
    at javax.media.j3d.ImageComponentRetained.createRenderedImageDataObject(ImageComponentRetained.java:815)
    at javax.media.j3d.ImageComponentRetained.createRenderedImageDataObject(ImageComponentRetained.java:849)
    at javax.media.j3d.ImageComponent2DRetained.set(ImageComponent2DRetained.java:142)
    at javax.media.j3d.ImageComponent2D.<init>(ImageComponent2D.java:205)
    at com.sun.j3d.utils.image.TextureLoader.getTexture(TextureLoader.java:544)
    at com.sun.j3d.loaders.objectfile.ObjectFileMaterials.readMapKd(ObjectFileMaterials.java:301)
    at com.sun.j3d.loaders.objectfile.ObjectFileMaterials.readFile(ObjectFileMaterials.java:353)
    at com.sun.j3d.loaders.objectfile.ObjectFileMaterials.readMaterialFile(ObjectFileMaterials.java:400)
    at com.sun.j3d.loaders.objectfile.ObjectFile.loadMaterialFile(ObjectFile.java:525)
    at com.sun.j3d.loaders.objectfile.ObjectFile.readFile(ObjectFile.java:589)
    at com.sun.j3d.loaders.objectfile.ObjectFile.load(ObjectFile.java:1248)
    at com.sun.j3d.loaders.objectfile.ObjectFile.load(ObjectFile.java:718)
    at PiezaBasica.<init>(PiezaBasica.java:24)
    at PiezaRotable.<init>(PiezaRotable.java:52)
    at Musculos.<init>(Musculos.java:363)
```

Figura 3-23 Ventana de error en Java al cargar texturas

Fuente: Autores

Solución de errores

Solución 01: Las imágenes de las texturas tienen un tamaño muy grande

Se deben eliminar las texturas “bump.jpg” y “specular.jpg” desde 3D Max usando la herramienta *Maps* de todas las capas. Mediante el uso de Adobe Photoshop se debe cortar y editar las texturas para ubicarlas individualmente en cada elemento “.obj” y su nuevo tamaño predeterminado será 1024x1024 pixeles.

Para confirmar que la prueba es la mejor; se verifica la correcta posición de los objetos y que las texturas se vean lo más enfocadas posible.

Se ha terminado la iteración HU1, cumpliendo con la tarea de la historia de usuario respectivo.

HU2: Creación de un aplicativo para manipular imágenes 3D

De acuerdo a la Tabla 3-6 Historia de Usuario HU2 previamente revisada se establecerá la estructura del sistema a ser utilizado durante el resto del proyecto para el aplicativo 3D.

Prototipo 1. Implementación del aplicativo con Java 3D

Se procede a la implementación de un aplicativo para la manipulación de los “.obj”, el cual será construido en Java junto a la librería Java 3D adecuada para la creación y manipulación de entornos 3D. Para lo cual se iniciará con el entorno de trabajo generado para el programa Java 3D.

A. Grafo de Escena

Es una estructura de datos compuesta por nodos y arcos y sirve como una especificación de documentación que permite representar gráficamente la información referente a geometría, sonido, luces, localización, orientación y apariencia de los objetos del ambiente virtual.

Símbolos utilizados para construir un grafo de escena.

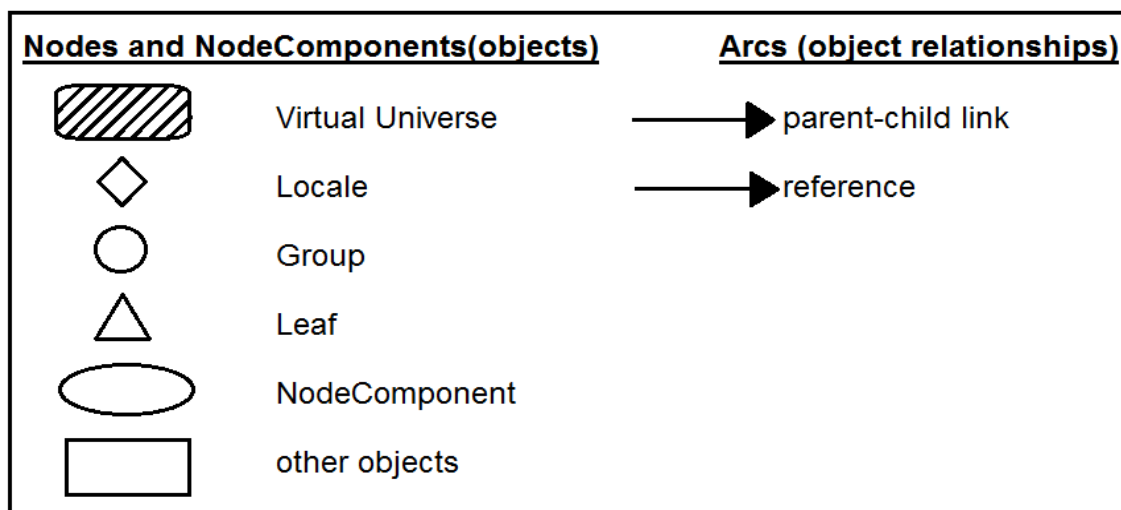


Figura 3-24 Símbolos que Representan Objetos en un Escenario Gráfico

Fuente : Tripod, Modelado Gráfico - Java 3D, 2012[74]

B. Tipos básicos de nodos

Existe una terminología y notación estandarizada de los componentes de un escenario gráfico en Java 3D, que se emplearán en la elaboración de este tipo de entorno para lo cual se debe comprender lo siguiente:

- a) **Nodo: (Node)** Un elemento en el escenario gráfico.
- b) **Nodo Hoja: (Leaf Node)** Representación de instancias de clase como un *Shape3D*, *Light*, *Behavior* y *Sound* derivadas de la propia clase *Leaf* son nodos que no tiene hijos.
- c) **Nodo Shape: (Shape Node)** Introduce objetos visibles dentro del mundo y posee dos campos:
 - **Appearance:** define el aspecto, la apariencia, usando las propiedades del objeto a crear.
 - **Geometry:** define de qué forma (geometría) será el objeto.
- d) **Nodo componente: (Node Component)** Es la superclase empleada para especificar las características de geometría, apariencia, sonido, textura y propiedades del material de un nodo "*Shape3D*" y puede ser referenciado por más de un objeto "*Shape3D*", siendo un conjunto de atributos para el nodo.
- e) **Nodos Grupo: (Group Node)** Representación de instancias de clases como *BranchGroup* y *TransformGroup*. Permite agrupar un conjunto de nodos y tratarlos como una sola entidad, son nodos con hijos.
 - El **BranchGroup** será el único tipo de hijo que pueda tener una Locación y cuando se adhiere a este se dice que se vuelve vivo, es decir, sus nodos están listos para ser renderizados.
 - El **TransformGroup** permite realizar transformaciones geométricas a todos los nodos que dependan del *Transform*.

- f) **Universo virtual: (Virtual Universe)** Puede ser una representación de instancia de un objeto de clase *VirtualUniverse* y representa el espacio virtual 3D con una colección de escenarios gráficos, cuenta con un solo universo por aplicación.
- g) **Locación: (Locale)** Describe un punto específico en el espacio 3D con una ubicación en el universo donde se coloca a los escenarios gráficos, contando con una locación por universo.
- h) **Rama gráfica: (Branch Graph)** Un escenario gráfico con varias ramas por locación.
- i) Los escenarios gráficos se divide en dos ramas gráficas.
 - **Rama de contenido: (Content Branch)** contiene formas, luces y otros contenidos, con varias ramas por locación.
 - **Rama de Visión. (View Branch)** es la información de visión y contiene una por escenario.

La Figura 3-25 muestra la estructura de un árbol para el sistema de ejecución Java 3D, el cual define un escenario gráfico que contiene un solo “*VirtualUniverse*”, un “*Locale*” para determinar la localización de los objetos visuales en el universo virtual, el “*BranchGroup*” que es la raíz del sub-gráfico o rama gráfica.

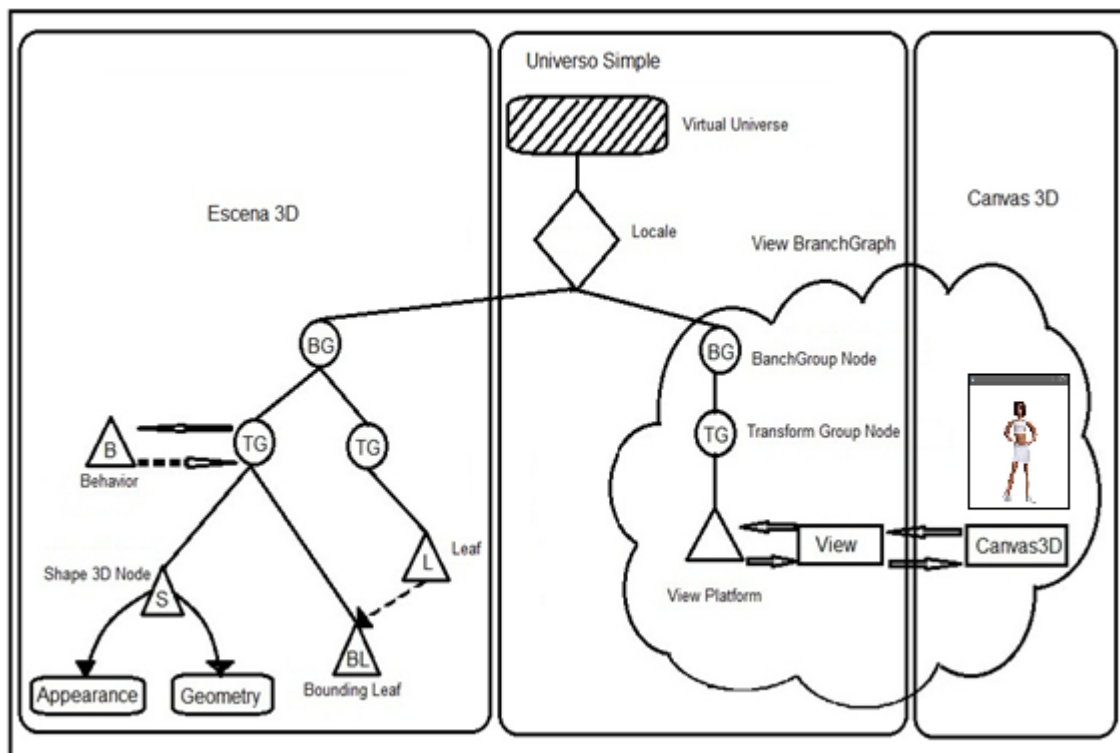


Figura 3-25 Escenario gráfico para el Applet 3D

Fuente: Autores

La clase “*SimpleUniverse*” es la que encapsulará los componentes “*VirtualUniverse*”, “*Locale*”, “*ViewPlatform*”, “*View*” y “*Canvas3D*” del espacio virtual 3D del aplicativo a desarrollar. Para la construcción del escenario gráfico se definió un solo *VirtualUniverse* el cual está asociado a un objeto *Locale* que determinará la localización de los objetos visuales en el universo virtual, donde este servirá de raíz para las dos ramas sub-gráficas que se generan, es decir, para la rama de vista gráfica y para la rama de contenido gráfico.

La rama de contenido gráfico especifica el contenido del universo virtual donde se ubicará la geometría, la apariencia, el comportamiento, la localización y las luces. Mientras que la rama de vista gráfica define los parámetros de visualización donde se ubicará la posición de visualización y la dirección.

Las dos ramas detallan la mayor parte del trabajo que el renderizador tiene que hacer.

La rama gráfica de *grupoBranchGroup* (BG) es el nodo que va a tener como hijos a los objetos de un escenario gráfico para ser renderizados en el cual, el grupo de transformación *TransformGroup* (TG) proporcionará la capacidad de realizar cualquier tipo de transformaciones que se requieran. En la rama de vista gráfica la plataforma de visión *View Platform* se usa para configurar la rama gráfica de un escenario y en conjunto con el objeto *Canvas3D* proporcionan una imagen en una ventana de la pantalla que mostrará la escena 3D. En la rama de contenido (escena3D) contendrá las formas 3D en las que se podrá realizar transformaciones (traslación, rotación, escala) a la posición en el espacio, donde el nodo hoja *Shape3D* (S) hará referencia a los nodos componente que especifican sus elementos geométricos *Geometry* y su apariencia *Appearance*, mientras la clase *Leaf* especifica la forma y comportamiento de los objetos visuales en el universo virtual, la clase *Behavior* especifica la animación o interacción con el objeto.

Como instancia final el universo simple será el que conecte la instancia *Canvas 3D* con la vista de escena 3D en el escenario gráfico.

C. Construcción de un programa en Java 3D

El uso de *SimpleUniverse* hace que el método básico sea sencillo. El método de elaboración del programa en Java 3D realizará los siguientes pasos:

- a) Primero se creará un objeto *Canvas3D*
- b) Se creará un objeto *SimpleUniverse* con referencias al objeto *Canvas3D*.
- c) Se configurará el objeto *SimpleUniverse*
- d) Se construirá el contenido de la escena (*Branch Graph*)
- e) Se compilará el contenido de la escena
- f) Se insertará el contenido de la escena en el objeto *Locale* del *SimpleUniverse*

El sistema 3D debe interactuar con el dispositivo de realidad virtual tomando los datos del dispositivo electrónico y del teclado de acuerdo a los datos que envíe para obtener una salida deseada.

Para la traslación y rotación en la escena 3D se deben leer los datos que el dispositivo electrónico envíe como cadena de caracteres, los cuales estarán asociados a valores (letras) en el teclado, la traslación se realiza en los ejes X e Y y la rotación será con respecto al eje Y. Además, el desplazamiento del puntero será en dos dimensiones (eje X e Y) que se obtendrá a partir de los movimientos del dispositivo y del mouse. Conjuntamente la acción de selección se la conseguirá al pasar el cursor sobre el objeto.

D. Casos de uso con notación UML:

A continuación se define la acción que existe entre el usuario o cliente mostrando el diagrama de uso general del sistema.

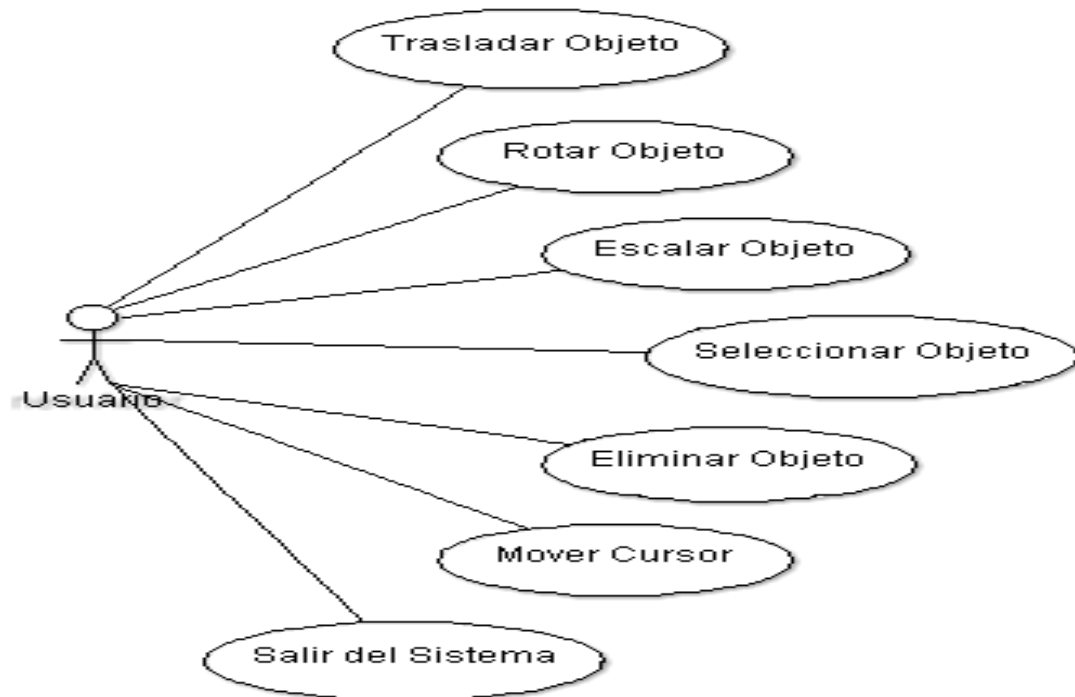


Figura 3-26 Diagrama casos de uso general del sistema

Fuente: Autores

Actor: Usuario

Actividad: Realiza el proceso de trasladar, rotar, escalar, seleccionar, eliminar el objeto 3D del universo virtual.

Observaciones: Al ingresar al Applet y realizar las trasformaciones con el mouse y el teclado para el objeto 3D, el proceso no estará activo hasta dar clic en el entorno del Applet.

El Applet de Java 3D puede ser acoplado en una página HTML por lo que se procede con su elaboración.

Diseño

Para el diseño de la aplicación que manipule objetos 3D se procede con la recomendación de la metodología ágil XP, trabajando en una iteración a la vez y evitando soluciones complicadas.

Iteración 1. Diseño e implementación del aplicativo con Java 3D

A. Diseño y diagrama de clases

El diseño de clases se basa en la orientación a objetos empleados en Java, se ha estructurado la aplicación en diferentes clases que se realizarán para el desarrollo del Applet.

Se estructura la solución en las siguientes clases:

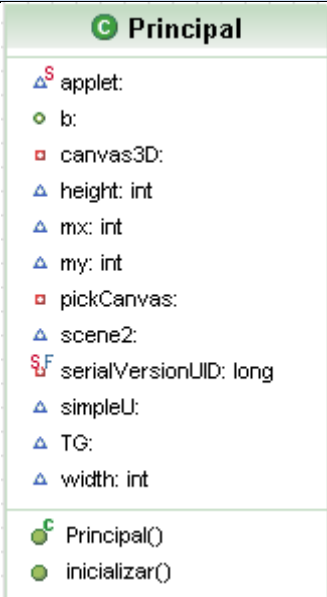
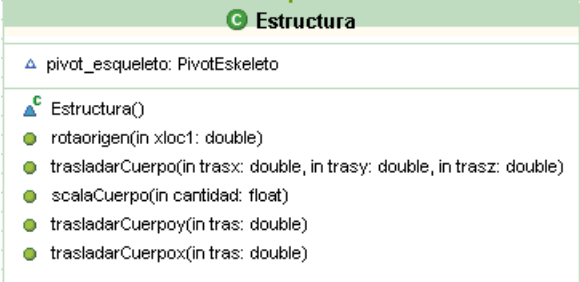
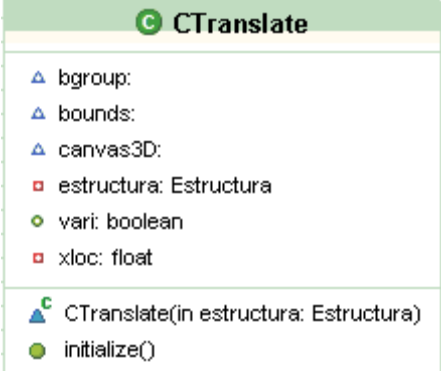
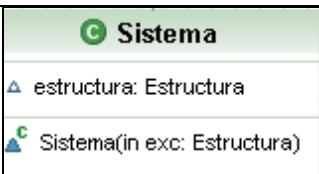
Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
Principal	Se encuentra el método que ejecuta la aplicación, contiene la estructura del Universo Virtual, la creación de escena y la llamada a los nodos hijos.	 <pre> classDiagram class Principal { applet: b: canvas3D: height: int mx: int my: int pickCanvas: scene2: serialVersionUID: long simpleU: TG: width: int Principal() inicializar() } </pre>
Estructura	Se enlaza con la clase PivotEsqueleto para incorporar los objetos a la escena	 <pre> classDiagram class Estructura { pivot_esqueleto: PivotEsqueleto Estructura() rotaorigen(in xloc1: double) trasladarCuerpo(in trasx: double, in trasy: double, in trasz: double) scalaCuerpo(in cantidad: float) trasladarCuerpoy(in tras: double) trasladarCuerpox(in tras: double) } </pre>
CTraslate	Maneja las funciones del teclado para la interacción con los objetos 3D, está relacionada con la clase Coordenadas.	 <pre> classDiagram class CTranslate { bgroup: bounds: canvas3D: estructura: Estructura vari: boolean xloc: float CTranslate(in estructura: Estructura) initialize() } </pre>
Sistema	Se enlaza con la clase estructura y coloca la textura de fondo	 <pre> classDiagram class Sistema { estructura: Estructura Sistema(in exc: Estructura) } </pre>

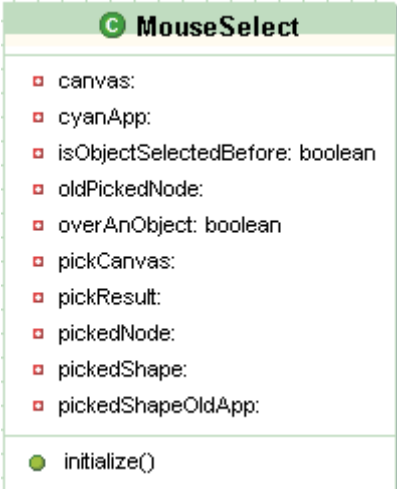
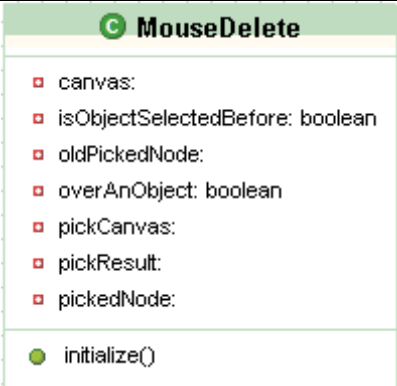
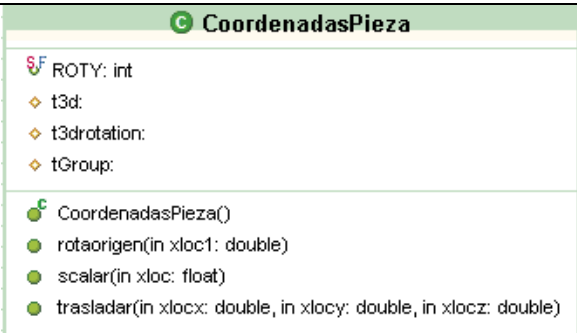
Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
MouseSelect	Selecciona los objetos al pasar por encima el mouse y lo coloreá de verde	 <pre> classDiagram class MouseSelect { canvas: cyanApp: isObjectSelectedBefore: boolean oldPickedNode: overAnObject: boolean pickCanvas: pickResult: pickedNode: pickedShape: pickedShapeOldApp: initialize() } </pre>
MouseDelete	Elimina el objeto que fue seleccionado con anterioridad	 <pre> classDiagram class MouseDelete { canvas: isObjectSelectedBefore: boolean oldPickedNode: overAnObject: boolean pickCanvas: pickResult: pickedNode: initialize() } </pre>
Coordenadas	Contiene las variables globales, las capacidades y los métodos relacionados con la transformación del objeto que se utilizarán para cada una de las geometrías creadas, manipuladas y adheridas a la escena gráfica.	 <pre> classDiagram class CoordenadasPieza { ROTY: int t3d: t3drotation: tGroup: CoordenadasPieza() rotaorigen(in xloc1: double) scalar(in xloc: float) trasladar(in xlocx: double, in xlocy: double, in xlocz: double) } </pre>

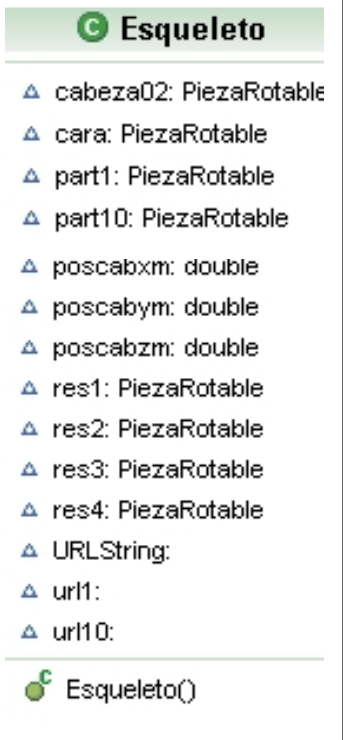
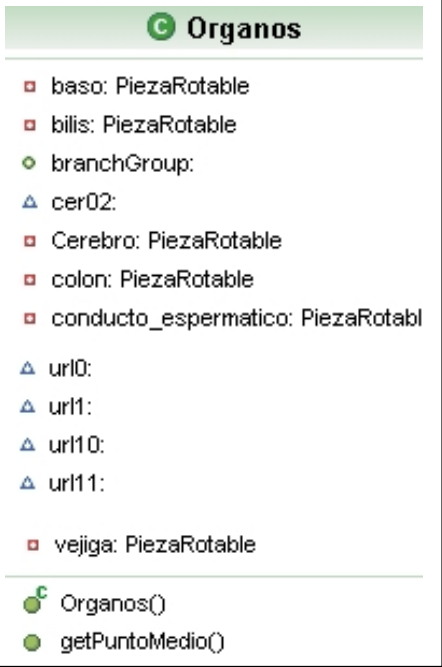
Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
Esqueleto	Contiene la estructura que llama a los objetos 3D para la capa de huesos desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la clase Coordenadas.	 <pre> classDiagram class Esqueleto { +cabeza02: PiezaRotable +cara: PiezaRotable +part1: PiezaRotable +part10: PiezaRotable +poscabxm: double +poscabym: double +poscabzm: double +res1: PiezaRotable +res2: PiezaRotable +res3: PiezaRotable +res4: PiezaRotable +URLString: +url1: +url10: } Esqueleto() </pre>
Órganos	Contiene la estructura que llama a los objetos 3D para la capa de órganos desde la URL y los adhiere a la escena cuando sean convocados, está relacionada con la clase Coordenadas.	 <pre> classDiagram class Organos { +baso: PiezaRotable +bilis: PiezaRotable +branchGroup: +cer02: +Cerebro: PiezaRotable +colon: PiezaRotable +conducto_espermatico: PiezaRotable +url0: +url1: +url10: +url11: +vejiga: PiezaRotable } Organos() getPuntoMedio() </pre>

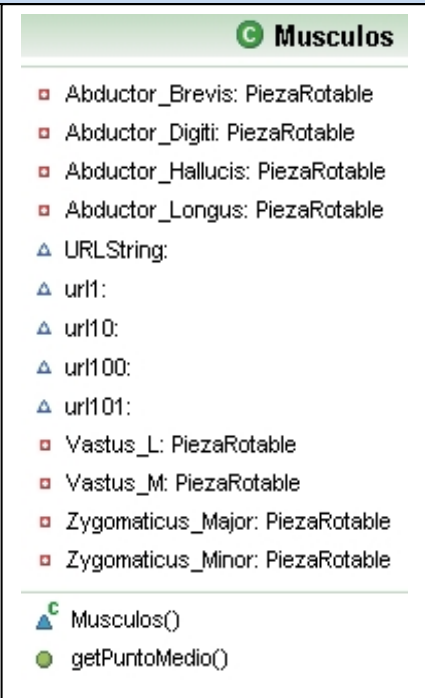
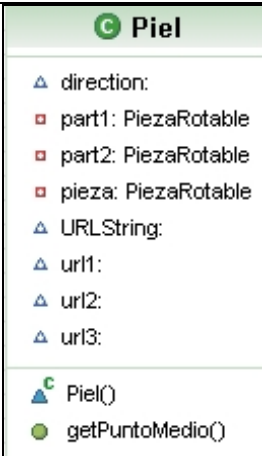
Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
Músculos	Contiene la estructura que llama a los objetos 3D para la capa de músculos desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la clase Coordenadas.	 <pre> classDiagram class Musculos { +Abductor_Brevis: PiezaRotable +Abductor_Digiti: PiezaRotable +Abductor_Hallucis: PiezaRotable +Abductor_Longus: PiezaRotable +URLString: +url1: +url10: +url100: +url101: +Vastus_L: PiezaRotable +Vastus_M: PiezaRotable +Zygomaticus_Major: PiezaRotable +Zygomaticus_Minor: PiezaRotable +Musculos() +getPuntoMedio() } </pre>
Piel	Contiene la estructura que llama a los objetos 3D para la capa piel desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la clase Coordenadas.	 <pre> classDiagram class Piel { +direction: +part1: PiezaRotable +part2: PiezaRotable +pieza: PiezaRotable +URLString: +url1: +url2: +url3: +Piel() +getPuntoMedio() } </pre>

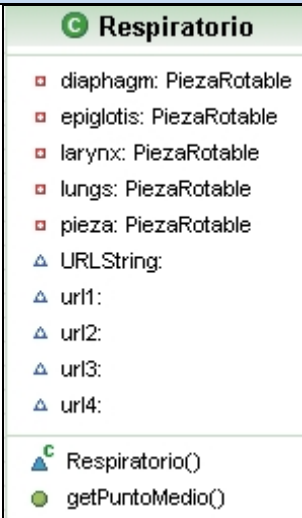
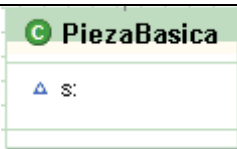
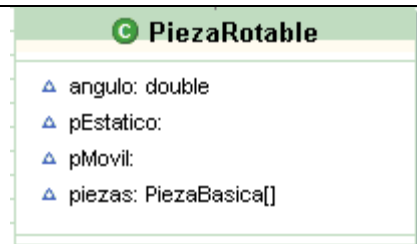
Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
SisRespiratorio	Contiene la estructura que llama a los objetos 3D para la capa del sistema respiratorio desde la URL y los adhiere a la escena cuando sean invocados, está relacionada con la clase Coordenadas.	 <pre> classDiagram class SisRespiratorio { +diaphragm: PiezaRotable +epiglotis: PiezaRotable +larynx: PiezaRotable +lungs: PiezaRotable +pieza: PiezaRotable +URLString: +url1: +url2: +url3: +url4: +Respiratorio() +getPuntoMedio() } </pre>
PiezaBásica	Encargada de la carga de los archivos .obj desde la URL y su posterior transformación a "Shape3D" para su manipulación, está relacionada con la clase Coordenadas.	 <pre> classDiagram class PiezaBasica { +s: } </pre>
PiezaRotable	Genera los métodos para realizar la rotación de los objetos 3D, está relacionada con la clase Coordenadas.	 <pre> classDiagram class PiezaRotable { +angulo: double +pEstatico: +pMovil: +piezas: PiezaBasica[] } </pre>

Diagrama de clases para el aplicativo 3D		
Clase	Función	Diagrama
Pivot Esqueleto	Mandar a llamar a cada una de las capas que contienen los objetos 3D y los adhiere a la escena, está relacionada con la clase Coordenadas.	<pre> class PivotEsqueleto { Δ intestino: Δ esqueleto: Esqueleto Δ lugar: Δ musculo: Musculos Δ organos: Organos Δ piel: Piel Δ pivot_principal: PiezaRotable Δ s: Δ sisrespiratorio: Respiratorio Δ uri: Δ corazon: C PivotEsqueleto() ● rotaorigen(in xloc1: double) ● scalar(in xloc: float) ● trasladar(in xlocx: double, in xlocy: double, in xlocz: double) ● trasladarx(in tras: double) ● trasladary(in tras: double) </pre>

Tabla 3-11 Diseño de Clases.

Fuente: Autores

El Diagrama de clases para el aplicativo 3D (ver Figura 3-27) se diseña con la librería Java 3D, donde la clase principal va a contener al *init()* que iniciará la aplicación y la integración de las clases para su posterior manipulación con el teclado, el mouse, la selección y eliminación del objeto y la clase coordenadas contendrá las transformaciones para las capas del cuerpo humano (ver anexo 5.2.1.9).

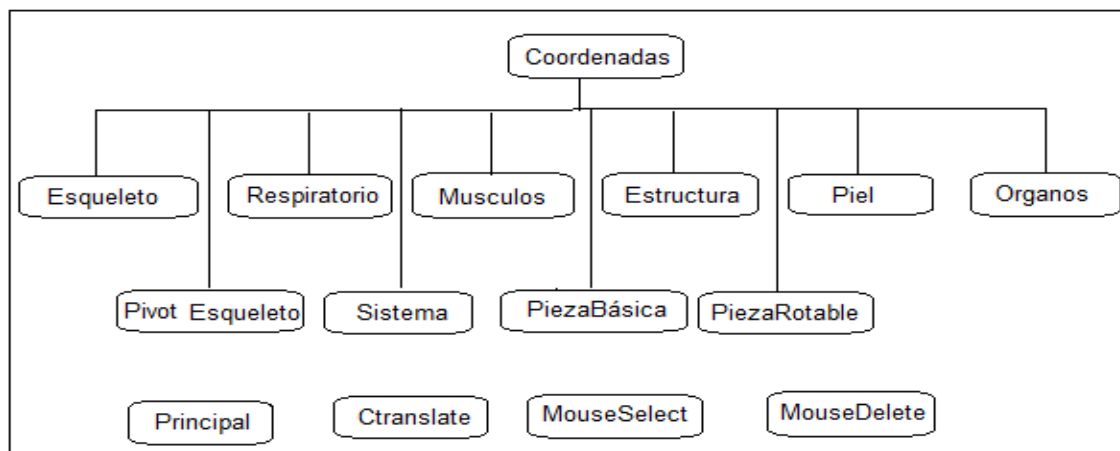


Figura 3-27 Diseño de Clases Java

Fuente: Autores

B. Descripción de librerías y capacidades

Para iniciar con la programación 3D se debe realizar enlaces a las librerías propias de Java 3D que servirán para elaborar el Applet. Con la biblioteca J3D se podrá hacer uso de los paquetes que se requieran para la elaboración del aplicativo 3D solo importando, de esta forma se podrá visualizar un objeto, su color y coordenadas:

Librerías J3D
<pre> import java.awt.*; import javax.media.j3d.*; import com.sun.j3d.utils.pickfast.PickCanvas; import com.sun.j3d.utils.universe.SimpleUniverse; import javax.vecmath.Color3f; //representa un color para un vértice import javax.vecmath.Point3d; //representa coordenadas de un vértice import javax.vecmath.Vector3d; //representa superficies normales en vértices import javax.vecmath.Vector3f; import java.applet.Applet; import java.awt.event.MouseEvent; import java.awt.event.MouseListener; import javax.media.j3d.BoundingSphere; import javax.media.j3d.BranchGroup; import javax.media.j3d.Canvas3D; import javax.media.j3d.PickInfo; import javax.media.j3d.Transform3D; import javax.media.j3d.TransformGroup; </pre>

Tabla 3-12 Librerías J3D para visualizar un objeto, su color y coordenadas

Fuente: Autores

De la lectura de ficheros de escena 3D y la creación de representaciones Java 3D se encarga el paquete `com.sun.j3d.loaders`, este puede convertir los ficheros creados en otras aplicaciones a Java 3D, y servirá para cargar los archivos `.obj`, también se incluyen la llamada de los objetos desde una URL:

Librerías J3D para cargar archivos .obj

```
import java.io.FileNotFoundException;
import com.sun.j3d.loaders.Scene; // Contiene los objetos cargados desde el Disco
import com.sun.j3d.loaders.objectfile.ObjectFile; // carga modelos .obj
import java.util.Hashtable;
import java.util.Enumeration;
import javax.media.j3d.*;
import java.net.*;
```

Tabla 3-13 Librerías J3D para cargar archivos .obj

Fuente: Autores

Para escribir la clase de comportamiento personalizado *Behavior* que actuará en el escenario gráfico con el movimiento del mouse y para la eliminación del objeto se hará referencia a lo siguiente.

Librerías J3D para comportamiento personalizado

```
import java.awt.Cursor;
import java.awt.event.MouseEvent;
import java.util.Enumeration;
import javax.media.j3d.Behavior;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.CapabilityNotSetException;
import javax.media.j3d.Node;
import javax.media.j3d.Shape3D;
import javax.media.j3d.WakeupCriterion;
import javax.media.j3d.WakeupOnAWTEvent;
import com.sun.j3d.utils.picking.PickCanvas;
import com.sun.j3d.utils.picking.PickResult;
```

Tabla 3-14 Librerías J3D para comportamiento personalizado

Fuente: Autores

Cuando una rama gráfica es compilada el sistema la convierte a una representación interna más eficiente mejorando el renderizado y fijando el valor de transformaciones y otros objetos en el escenario gráfico.

Como cada “*SceneGraphObject*” tiene un conjunto de bits de capacidad, los valores de estos bits determinarán que capacidades deben estar activas antes de compilarlo o darle vida para la manipulación del objeto, ya que si no se lo especifica en el programa no podrán cambiar los valores de transformación del objeto en el escenario una vez que estén vivos, por lo cual la Tabla 3-15 muestra los datos necesarios para que el objeto pueda ser transformado:

Capacidades de TransformGroup	
Detalle	Lista de capacidades
Permite acceder y escribir la información de transformación de su objeto	tGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_READ); tGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

Tabla 3-15 Lista de capacidades de TransformGroup

Fuente: Autores

También controlan el acceso a otros aspectos de un objeto “*TransformGroup*”, los cuales heredan configuraciones de capacidades de sus clases ancestros como: “*Group*” y “*Node*”.

Capacidades de TransformGroup	
Detalle	Lista de capacidades
Permite añadir hijos al nodo <i>Group</i> después de compilado	- (BranchGroup.ALLOW_CHILDREN_EXTEND);
Permite leer las referencias a los hijos del nodo <i>Group</i> después de compilado	- (TransformGroup.ALLOW_CHILDREN_READ);
Permite rescribir las referencias a los hijos del nodo <i>Group</i> después de compilado	- (BranchGroup.ALLOW_CHILDREN_WRITE);
Permite leer o escribir su	-(TransformGroup.ALLOW_PICKABLE_READ);

Capacidades de TransformGroup	
Detalle	Lista de capacidades
estado de selección al dar clic en el objeto.	-(TransformGroup.ALLOW_PICKABLE_WRITE); -(BranchGroup.ALLOW_PICKABLE_READ);
Especifica que este nodo será reportado en el “SceneGraphPath” si ocurre una selección o una colisión	-(BranchGroup.ENABLE_PICK_REPORTING); - (TransformGroup.ENABLE_PICK_REPORTING) - (TransformGroup.ENABLE_COLLISION_REPORTING); - (TransformGroup.ALLOW_LOCAL_TO_VWORLD_READ);
Proporciona la capacidad para comprobar la intersección de un objeto “PckShape” y geometrías primitivas	-(Geometry.ALLOW_INTERSECT); -(BranchGroup.ALLOW_DETACH);
Los objetos “Shape3D” también heredan capacidades de las clases SceneGraphObject, Node, y Leaf.	- TransformGroup.ALLOW_COLLISION_BOUNDS_WRITE);

Tabla 3-16 Lista de capacidades de otros aspectos del TransformGroup

Fuente: Autores

Codificación

Iteración 1. Desarrollo e implementación del aplicativo con Java 3D

Capa: Piel



Figura 3-28 Ventana de la capa de piel

Fuente: Autores

Módulo: Capa piel

Pantalla: Applet capa piel

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa de piel será la primera en ser visualizada y retirada al seleccionar y dar clic.

Parte del código es descrito a continuación:

Se crean variables que enlacen con la clase *PiezaRotable*, se coloca en una cadena de caracteres la dirección (URL) de la carpeta de donde serán leídas las imágenes 3D.

El constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:

```

Piel() {
    try {
url1=newURL(URLString+"piel/Model_BodyGenitales.obj");
        part1 = newPiezaRotable(url1);//piel
        part1.trasladarHacia(newVector3d(0,0,0));
//Adhiere el objetohijo
tGroup.addChild(part1.tGroup);

```

Capa: Músculos

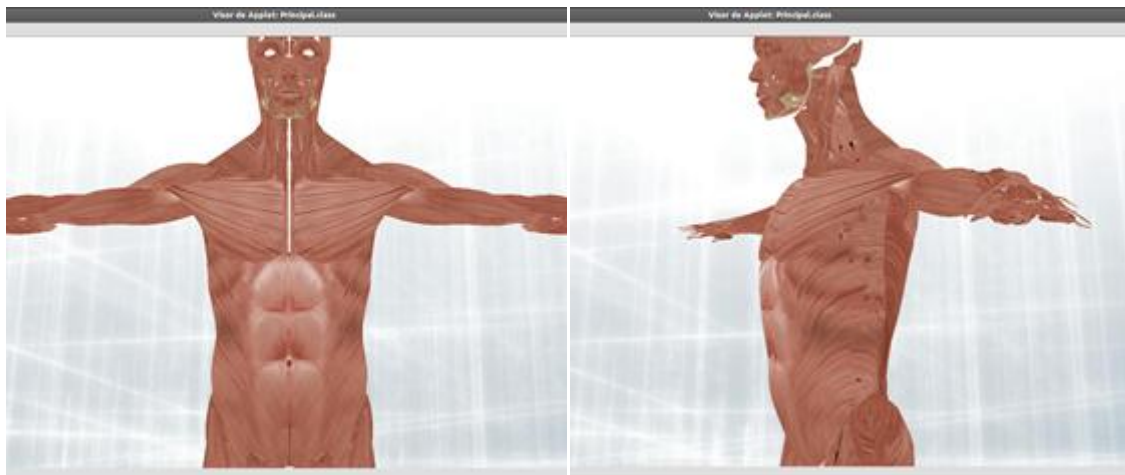


Figura 3-29 Ventana de la capa músculos

Fuente: Autores

Módulo: Capa músculos

Pantalla: Applet capa músculos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: posicionar puntero del mouse y clic

Descripción: La capa de músculos será la siguiente en ser visualizada después de retirar la capa piel y cada objeto elegido se retirará al seleccionar y dar clic.

Parte del código es descrito a continuación:

Se crearán variables que enlacen con la clase PiezaRotable, se coloca en una cadena de caracteres la dirección (URL) de la carpeta de donde serán leídas las imágenes 3D.

El constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:

```
Musculos() {
    try {
        url1=newURL(URLString+"Model_Muscular/Model_Muscular_Adductor_Brevis.obj");
        Abductor_Brevis.trasladarHacia(newVector3d(0,0,0));
        Abductor_Brevis = newPiezaRotable(url1);
        //Adhiere el objetohijo
        tGroup.addChild(Abductor_Brevis.tGroup);
    }
```

Capa: Órganos



Figura 3-30 Ventana de la capa órganos

Fuente: Autores

Módulo: Capa órganos

Pantalla: Applet capa órganos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa de órganos será la siguiente en ser visualizada después de retirar algunos objetos de la capa músculos y cada elemento elegido se retirará al seleccionar y dar clic.

Parte del código es descrito a continuación:

Se crearán variables que enlacen con la clase PiezaRotable, se coloca en una cadena de caracteres la dirección (URL) de la carpeta de donde serán leídas las imágenes 3D.

El constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:

```
Organos() {  
    try {  
        url1=newURL(URLString+"Model_Organ/esofago.obj");  
        esofago.trasladarHacia(newVector3d(0,0,0));  
        esofago = newPiezaRotable(url1);  
        //Aadhiere el objeto hijo  
        tGroup.addChild(esofago.tGroup);  
    }
```

Capa: Sistema Respiratorio



Figura 3-31 Ventana de la capa sistema respiratorio

Fuente: Autores

Módulo: Capa Sistema Respiratorio

Pantalla: Applet capa Sistema Respiratorio

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa del sistema respiratorio será la siguiente en ser visualizada después de retirar algunos objetos de la capa músculos y cada elemento elegido se retirará al seleccionar y dar clic.

Parte del código es descrito a continuación:

Se crearán variables que se enlacen con la clase `PiezaRotable`, se coloca en una cadena de caracteres la dirección (URL) de la carpeta de donde serán leídas las imágenes 3D.

El constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:


```

Organos() {
    try {
        url1=newURL(URLString+"Model_ Respiratory
/Model_Respiratory_Diaphragm.obj");
        diaphragm.trasladarHacia(newVector3d(0,0,0));
        diaphragm = newPiezaRotable(url1);
//Aadhiere el objeto hijo
tGroup.addChild(diaphragm.tGroup);

```

Capa: Esqueleto

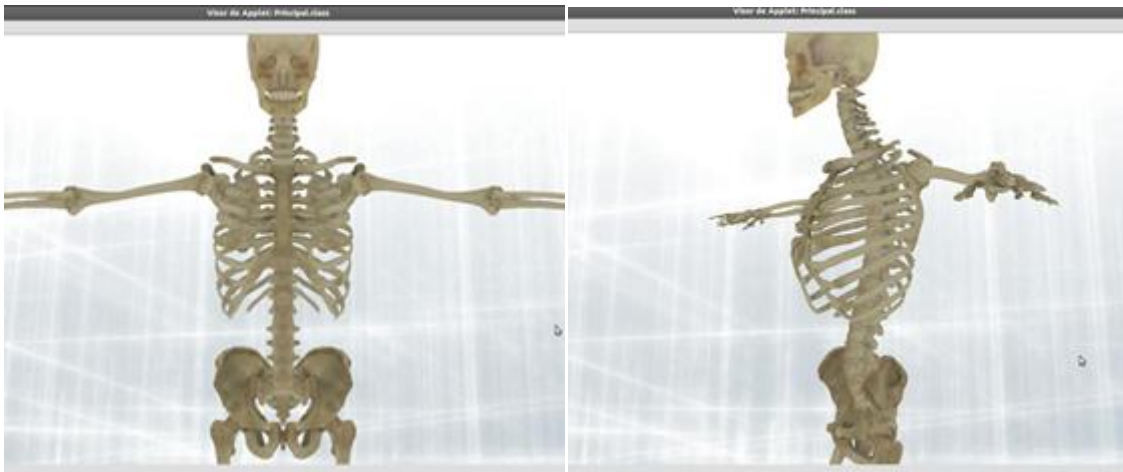


Figura 3-32 Ventana de la capa esqueleto

Fuente: Autores

Módulo: Capa esqueleto

Pantalla: Applet capa esqueleto

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa esqueleto será la siguiente en ser visualizada después de retirar algunos objetos de las capas de: músculos y órganos, cada elemento elegido se retirará al seleccionar y dar clic.

Parte del código es descrito a continuación:

Se crearán variables que se enlacen con la clase `PiezaRotable`, se coloca en una cadena de caracteres la dirección (URL) de la carpeta de donde serán leídas las imágenes 3D.

El constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:

```
Esqueleto () {
    try {
        url1 = newURL(URLString+"Model_Skeletal/Metacarpal.obj");
        part1 = newPiezaRotable(url1);
        part1.trasladarHacia(newVector3d(poscabxm, poscabym, poscabzm));
        //Se adhiere el objeto hijo
        tGroup.addChild(part1.tGroup);
    }
```

Capa: Selección de Objetos

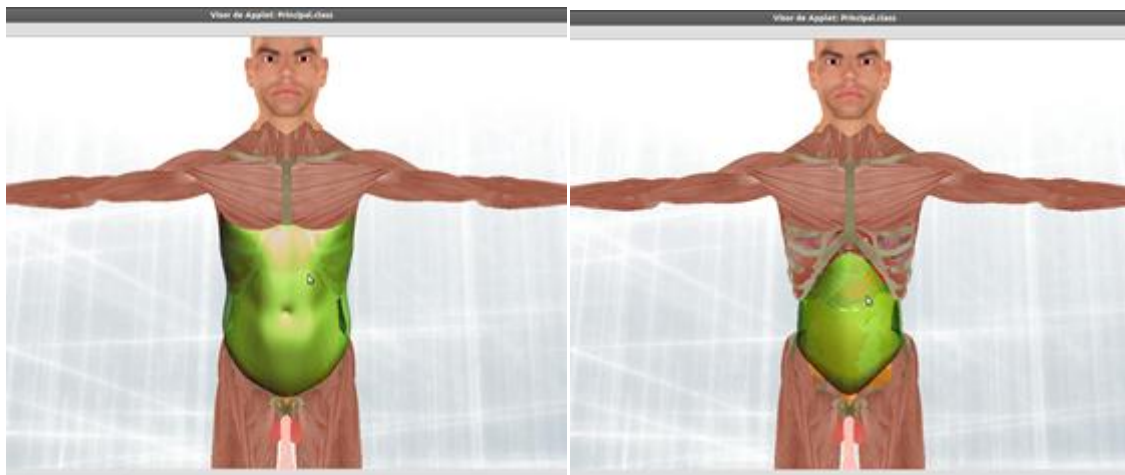


Figura 3-33 Ventana de la capa de selección de objetos

Fuente: Autores

Módulo: Capa de selección de objetos

Pantalla: Applet capa de selección de objetos

Proceso: Ejecuta el código al posicionar el puntero del mouse en el escenario gráfico

Evento: Posicionar puntero del mouse

Descripción: La capa de selección de objetos actuará en el momento en que el cursor del mouse se posicione sobre cualquier objeto, con lo que inmediatamente se tornará de un color verdoso estando listo para realizar cualquier acción posterior al dar clic.

Utilizando la librería *behavior* propia de Java se creará un comportamiento que espere eventos del ratón en el escenario gráfico para ejecutar el código específico del objeto seleccionado.

Parte del código es descrito a continuación:

En el constructor se recibe el *Canvas3D* y el *BranchGroup* provenientes de la clase principal de donde se envían los parámetros para que interactúe con el objeto al pasar el cursor

Permite al objeto la capacidad que al interceptarse con el mouse reaccione a un evento

Cambia el color del objeto al pasar el mouse y lo hace transparente

Llama al método cuando el *behaviour* es insertado dentro de la escena

El método *processStimulus* es llamado cuando el evento es retenido del segmento, especificando el movimiento del criterio de llegada.

```

public MouseSelect (Canvas3D canvas, BranchGroup dataBranchGroup) {
cyanApp.setTransparencyAttributes (newTransparencyAttributes
(TransparencyAttributes.NICEST, 0.3f));
}

public void initialize () {
    wakeupOn (newWakeupOnAWTEvent (MouseEvent.MOUSE_MOVED));
}

public void processStimulus(Enumeration criteria) {...}
...
if (pickResult != null) {//si el resultado de seleccionar es un objeto
pickedNode = ((Shape3D) pickResult.getNode (PickResult.SHAPE3D));
pickedShape = ((Shape3D) pickedNode);
//si al cruzar el puntero en la escena ocurre la intersección con el objeto cambie de color
if (isObjectSelectedBefore) {
    if (oldPickedNode == null) {...}
    else {
        ((Shape3D) oldPickedNode).setAppearance (pickedShapeOldApp);
        oldPickedNode = pickedShape;
        ...}
    }
}
isObjectSelectedBefore = true;

```

Capa: Eliminación de Objetos

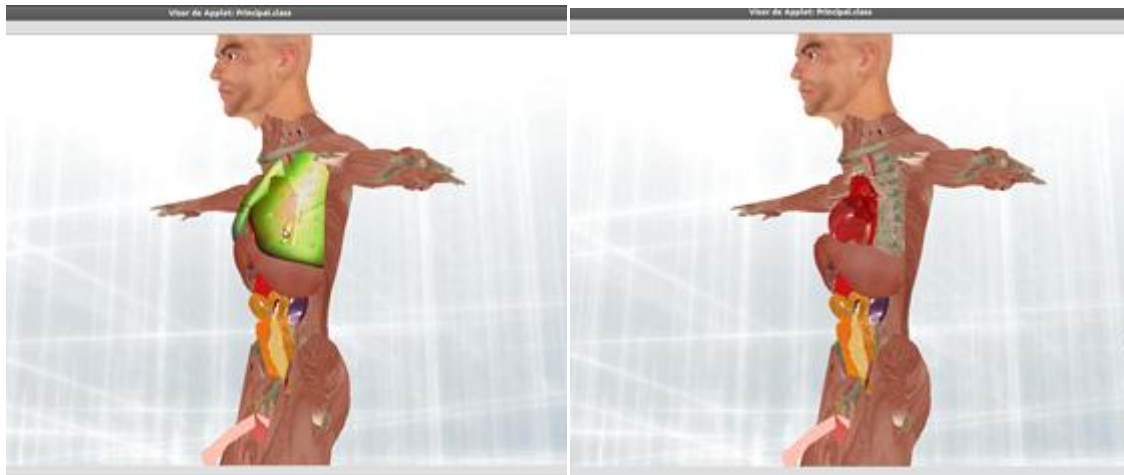


Figura 3-34 Ventana de la capa de eliminar objetos

Fuente: Autores

Módulo: Capa de eliminar objetos

Pantalla: Applet capa de eliminar objetos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar y clic con el puntero del mouse

Descripción: La capa de eliminar objetos actuará en el momento en que el puntero del mouse sea presionado sobre cualquier objeto luego de haber sido posicionado, seleccionado y puesto de color verde, con lo que inmediatamente será quitado de la escena, quedando listo para la siguiente acción.

Establece un comportamiento que espera eventos del ratón en el escenario gráfico.

La interacción se construirá a partir de la clase “*behavior*” propia de Java, que se utilizará en la eliminación del objeto.

Parte del código es descrito a continuación:

Una subclase de *PickTool*, simplifica la recogida de datos usando los eventos del ratón a partir de un lienzo

Esta clase permite escoger el uso de localizaciones lienzo x,y mediante la generación de la forma apropiada de recolección.

```
pickCanvas = newPickCanvas (canvas, dataBranchGroup);
```

Posteriormente llama al método cuando el *Behaviour* es insertado dentro de la escena y el método *processStimulus* es llamado cuando el evento es retenido del segmento, especificando el movimiento del criterio de llegada.

Se utiliza una Bandera *SHAPE3D* para pasar a *GetNode* (int) para devolver un nodo *Shape3D* de la *SceneGraphPath* y si al cruzar el puntero en la escena ocurre la intersección con el objeto se eliminará.

```
public MouseDelete (Canvas3D canvas, BranchGroup dataBranchGroup) {
    this.canvas = canvas;
    pickCanvas = newPickCanvas (canvas, dataBranchGroup);
}

public void initialize () {
    wakeupOn (newWakeupOnAWTEvent (MouseEvent.MOUSE_PRESSED));
}

public void processStimulus(Enumeration criteria) {...}
...
if (pickResult != null) { //si hay una imagen seleccionada
    pickedNode = ((Shape3D) pickResult.getNode (PickResult.SHAPE3D)); //Bandera.
    Shape3D shape = ((Shape3D) pickedNode);
    if (isObjectSelectedBefore) { //si ocurre la intersección
        if (oldPickedNode == null) {...}
    }
    else {
        canvas.setCursor (Cursor.getPredefinedCursor
        (Cursor.CROSSHAIR_CURSOR));
    }
}
```

Si el objeto seleccionado es un objeto se van removiendo todas las geometrías que tienen los *shape* al dar clic.

```

    if (shape != null) {
        shape.removeAllGeometries();
    }
    isObjectSelectedBefore = true;
}
wakeupOn (newWakeupOnAWTEvent (MouseEvent.MOUSE_PRESSED));

```

Capa: Estructura

Módulo: Estructura del nodo hijo

Proceso: Ejecuta el código al ser llamado en la instancia de la clase Sistema.

Descripción: La capa estructura será la que adhiera a la escena al constructor *pivot_esqueleto* que contiene a los nodos hijos que serán visualizados.

Parte del código es descrito a continuación:

Se creará la variable que se enlace con la clase *PivotEsqueleto* y en el constructor se crea la instancia a la clase *pivot_esqueleto* adhiriendo al nodo hijo.

Posteriormente se adhieren los métodos de rotar, escalar, trasladar al nodo hijo y enviar a su posición original.

```

Estructura () {
    this.pivot_esqueleto = newPivotEsqueleto();
    tGroup.addChild(pivot_esqueleto.tGroup);    }
    public void scalaCuerpo(float cantidad) { //zoom de la imagen
        pivot_esqueleto.scalar(cantidad);    }
    public void trasladarCuerpoy(double tras) { //traslada la imagen en Y
        pivot_esqueleto.trasladary(tras);    }
}

```

```

public void trasladarCuerpox(double tras) {//traslada la imagen en X
    pivot_esqueleto.trasladarx(tras);    }
public void trasladarCuerpo(double trasx,double trasy,double trasz) {//traslada al origen
    pivot_esqueleto.trasladar(trasx,trasy,trasz); }
public void rotaorigen(double xloc1){//rotación
    pivot_esqueleto.rotaorigen(xloc1); }

```

Capa: Sistema

Módulo: Sistema con nodos hijos

Proceso: Ejecuta código al cargar el sistema principal en la creación de la escena.

Descripción: La capa sistema será la que llame a los nodos hijos mediante la clase estructura y los adhiera a la escena para visualizar, también contiene el fondo de pantalla.

Parte del código es descrito a continuación:

Crear la variable que se enlaza con la clase Estructura y en el constructor crear la instancia a dicha clase y lo adhiere al nodo hijo.

Crear los límites para el *background* y *behaviors*

```

Sistema(Estructura exc) {
    super();
    estructura = exc;
    tGroup.addChild(exc.tGroup);
    BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 1000.0);

```

Llama a la textura para el fondo de pantalla, proveniente de una carpeta local y la adhiere a la escena


```

URL texturaBackground = this.getClass().getResource("texturas/fondo.jpg");
TextureLoaderbgTexture = newTextureLoader(texturaBackground, Principal.applet);
Background bg = newBackground(bgTexture.getImage());
bg.setApplicationBounds(bounds);
bg.setImageScaleMode(Background.SCALE_FIT_MAX);
tGroup.addChild(bg); }

```

Capa: PivotEsqueleto

Módulo: Sistema eje para la estructura de los objetos

Proceso: Ejecuta código al cargar el sistema CTranslate en la creación de la escena.

Descripción: La clase PivotEsqueleto será la que conecte a los nodos hijos al eje y los adhiera a la escena para visualizar.

Parte del código es descrito a continuación:

Crear variables que se enlazan con las clases que contienen las partes del cuerpo humano y en el constructor de la clase llama a las imágenes desde la URL especificada e indica las coordenadas en las que aparecerá el objeto, se genera una excepción para la URL y posteriormente adherir el objeto hijo (child) al grupo:

```

Esqueleto () {
url = new URL("http://190.15.136.20/img_cuerpo3D/Model_Skeletal/mandiblee1.obj");
pivot_principal = newPiezaRotable(url, newPoint3d(0,0,0), newPoint3d(0,0,0));
tGroup.addChild(pivot_principal.tGroup);

```

Crea las instancias a todas las clases que contienen los objetos a adjuntar y adhiere a los objetos hijos al eje principal posteriormente se llama a los métodos para que realicen zoom, traslación y rotación de la imagen:

```

pivot_principal.tGroup.addChild(piel.tGroup); //piel
public void scalar(float xloc) {
    pivot_principal.scalar(xloc); }
public void trasladary(double tras) {
    pivot_principal.trasladarHacia(new Vector3d(0, tras, 0)); }
public void trasladarx(double tras) {
    pivot_principal.trasladarHacia(new Vector3d(tras, 0, 0)); }
public void trasladar(double xlocx, double xlocy, double xlocz) {
    pivot_principal.trasladar(xlocx, xlocy, xlocz); }
public void rotaorigen(double xloc1) {
    pivot_principal.rotaorigen(xloc1); }

```

Capa: PiezaRotable

Módulo: Sistema que posiciona al objeto proveniente de la URL

Proceso: Ejecuta código al posicionar los objetos en la creación de la escena.

Descripción: La clase PiezaRotable será la que posicione a los nodos hijos en la escena para visualizar.

Parte del código es descrito a continuación:

Crear variables tipo *Point3d* y variables que se enlazan con la clase PiezaBásica

```

Point3dpMovil;
Point3dpEstatico;
double angulo;
PiezaBasica[] piezas;

```

En el constructor crea la instancia a la clase PiezaBásica y adhiere la posición al nodo hijo en el eje x, y, z

```
publicPiezaRotable(URL url, Point3dpMovil2, Point3dpEstatico2) { super();
piezas = newPiezaBasica[1];
piezas[0] = newPiezaBasica(url);
tGroup.addChild(piezas[0].tGroup); }
```

En el constructor crea la instancia a la clase PiezaBásica y adhiere la posición al nodo hijo en la posición por defecto

```
publicPiezaRotable(URL url) { super();
piezas = newPiezaBasica[1];
piezas[0] = newPiezaBasica(url);
tGroup.addChild(piezas[0].tGroup); }
```

Capa: PiezaBásica

Módulo: Sistema Pieza Básica

Proceso: Ejecuta código al cargar el sistema principal en la creación de la escena.

Descripción: La clase PiezaBásica será quien haga la llamada al URL y adhiera a los nodos hijos a la escena para visualizar.

Parte del código es descrito a continuación:

El constructor establece la instancia a la clase estructura y adhiere al nodo hijo, también introduce la variable *scena* y en el método PiezaBásica crea la instancia para cargar el archivo

```
Scene s;
publicPiezaBasica(URL url2){ super();
ObjectFile file = newObjectFile(ObjectFile.TRIANGULATE);
s = file.load(url2);
tGroup.addChild(s.getSceneGroup());//se adhiere a la escena
```

Los objetos *hashtable* son usados como *keys* o como *value*, para almacenar y recibir objetos; se crea la clase *hashtable* con la referencia *allReferences* para obtener todos los *keys* almacenados en *hashtable*, usar *enumeration* y mientras tenga elementos se adherirán

```
Hashtable allReferences = s.getNamedObjects();
Enumeration enumeration = allReferences.keys();
while (enumeration.hasMoreElements()) {
    String key = (String) enumeration.nextElement();
    if (allReferences.get(key) instanceof Shape3D) {
        Shape3D ref = (Shape3D) allReferences.get(key);
```

Envía las capacidades que son necesarias para adherir y remover

```
        ref.setCapability(Shape3D.ALLOW_APPEARANCE_OVERRIDE_WRITE);
        ref.setCapability(Shape3D.ALLOW_APPEARANCE_WRITE);
        ...
    } }
```

Capa: CTranslate

Módulo: Sistema para trasladar los nodos hijos con el teclado

Proceso: Ejecuta código al cargar el sistema principal en la creación de la escena.

Descripción: La clase CTranslate actuará en el momento que se haga uso del teclado, las acciones que realizará con los objetos serán las de: trasladar en los ejes X e Y, rotar (horario y anti-horario), escalar (zoom) y volver a su posición original.

La interacción se construirá a partir de la clase *Behavior* propia de Java, que incluye eventos del teclado, para la manipulación del objeto.

Parte del código es descrito a continuación:

Crear la variable que se enlacen con la clase Estructura.

En el constructor crear la instancia a la clase estructura y adherir al nodo hijo

Crear los límites para el *background* y *behaviors*

```
CTranslate(Estructuraestructura) {
this.estructura = estructura;    }
public void initialize() {
this.wakeupOn(new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED));  }
public void processStimulus(Enumeration criteria) {...}
...
}
```

El *behavior* permitirá la manipulación de los objetos dentro de la escena con los eventos del teclado, la siguiente tabla muestra el detalle que hará cada tecla al ser presionada.

Eventos del teclado	
Evento	Detalle
KEY_O	Vuelveel grupo de objetos al origen en la posicion 0
KEY_Left	Traslada en forma horizontal en el eje negativo
KEY_Right	Traslada en forma horizontal en el eje positivo
KEY_Down	Traslada en forma vertical en el eje negativo
KEY_Up	Traslada en forma vertical en el eje positivo
KEY_D	Rotaen el eje X positivo
KEY_A	Rotaen el eje X negativo
KEY_E	Realizael aumento delzoom en el eje Y
KEY_Q	Realizala disminución del zoom en el eje Y

Tabla 3-17 Eventos del teclado para manipulación de los objetos 3D

Fuente: Autores

Al efectuar un evento en el teclado con las teclas establecidas se realizará la acción de rotar y desplazarse en el eje X e Y, además cada vez que sea presionada la tecla establecida se hará un aumento o disminución en el tamaño del objeto en 10, es decir, será como si se hubiese dado 10 pulsaciones seguidas con lo cual se podrá

apreciar de mejor manera el zoom, estableciendo también un tamaño mínimo y máximo para que no se deforme el objeto.

```
switch (ev.getKeyCode()) {
case KeyEvent.VK_D://rotación
estructura.rotaorigen(-Math.PI / 6);
break;
case KeyEvent.VK_A:
...
break;
case KeyEvent.VK_DOWN://traslación
    estructura.trasladarCabinay((Math.PI / 32)-10.0);
break;
case KeyEvent.VK_UP: ...//cada evento aumentará 10pulsos
break;
case KeyEvent.VK_LEFT: ...// cada evento aumentará 10pulsos
break;
case KeyEvent.VK_RIGHT: ...//cada evento aumentará 10pulsos
break;
case KeyEvent.VK_O:
...
break;
//scale-zoom
case KeyEvent.VK_Q://1.0 tamaño mínimo
if(xloc>=1.0f&&xloc<=7.1f){ ...}//paracada1.0f será necesariodar 10 clic
break;
case KeyEvent.VK_E://13.1 tamaño máximo para zoom
    if(xloc>=1.1f&&xloc<=7.2f){...}
    break;    }}
this.wakeupOn(new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED)); }
```

Capa: CoordenadasPieza**Módulo:** Sistema coordenadas de los objetos**Proceso:** Ejecuta código al cargar el sistema.**Descripción:** En la clase CoordenadasPieza se declaran las capacidades para manipular los elementos, contiene los métodos para trasladar, rotar y escalar un objeto.

Parte del código es descrito a continuación:

Crear las variables que se enlacen con la clase Estructura.

En el constructor envía las capacidades que son necesarios para adherir, remover, pinchar (clic), interceptar, escribir, leer y los necesarios para realizar la transformación de los objetos.

```

public CoordenadasPieza() {
    t3d = new Transform3D();
    t3drotation = new Transform3D(); //Transform3Dpararotarlasimágenes
    tGroup = new TransformGroup();
    tGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    tGroup.setCapability(BranchGroup.ENABLE_PICK_REPORTING);
    tGroup.setCapability(Geometry.ALLOW_INTERSECT);
    tGroup.setCapability(BranchGroup.ALLOW_CHILDREN_WRITE);
    tGroup.setCapability(BranchGroup.ALLOW_DETACH);
    ...
}

```

Realiza los métodos para escalar, trasladara su posición original, rota las imágenes en torno al eje de coordenadas

```

public void scalar(float xloc){//Zoom
    t3d.setScale(new Vector3d(xloc, xloc, xloc));
    tGroup.setTransform(t3d); }

public void trasladar(double xlocx, double xlocy, double xlocz){// Trasladar
    t3d.setTranslation(new Vector3d(xlocx, xlocy, xlocz));
    tGroup.setTransform(t3d); }

public void rotaorigen(double xloc1){//Rotar
    t3drotation.rotY(xloc1);//ángulo
    t3d.mul(t3drotation);
    tGroup.setTransform(t3d);}

```

Capa: Principal

Módulo: Sistema Principal

Proceso: Ejecuta el inicio del sistema principal para la creación de la escena.

Descripción: La clase principal será la que inicialice el sistema mediante la creación del universo virtual y la llamada a escena al *BranchGroup*.

Parte del código es descrito a continuación:

Crea el *BranchGroup* para ser adherido al iniciar la escena

Crear los límites para el *background* y *behaviors*

```

public BranchGroup createSceneGraph(SimpleUniverses su) {
    BoundingSphere bounds = new BoundingSphere(new Point3d(), 100000.0);
    b = new BranchGroup();
    System.out.print("Cargando ");
    Estructura estructura = new Estructura();
    Sistema sistema = new Sistema(estructura);

```


La dirección de la luz es adherida a la raíz de la escena

```
DirectionalLight lightD1 = new DirectionalLight();
lightD1.setInfluencingBounds(bounds);
lightD1.setDirection(new Vector3f(-.3f, -.8f, -1f));
Transform3D tt = new Transform3D();
tt.setScale(0.02);
    TransformGroup tgt = new TransformGroup(tt);
b.addChild(tgt);
```

Arreglo de fondo de la imagen y mando a llamar las clases donde se encuentra la invocación al objeto y el control del teclado para adherirlos a la escena

```
Color3f bgColor = new Color3f(0.05f, 0.05f, 0.2f);
    Background bg = new Background(bgColor);
bg.setApplicationBounds(bounds);
tgt.addChild(bg);
tgt.addChild(sistema.tGroup); // llamo a métodos
CTranslate com = new CTranslate(estructura);
com.setSchedulingBounds(new BoundingSphere());
tgt.addChild(com);
tgt.addChild(lightD1);
```

Este método es llamado cuando ocurre un evento como el de seleccionar el objeto

```
MouseSelect behavior = new MouseSelect (canvas3D,b);
behavior.setSchedulingBounds (new BoundingSphere (new Point3d (), 100));
b.addChild (behavior);
```

Este método es llamado cuando ocurre el evento eliminar objeto por objeto, además se debe establecer la posición original 0,-100,-300 en donde el cuerpo humano va a aparecer y de esta manera sea apreciable en el entorno.

```
MouseDeletebehavior2 = new MouseDelete(canvas3D,b);
behavior2.setSchedulingBounds (new BoundingSphere (new Point3d (), 100));
b.addChild (behavior2);
b.compile();
sistema.trasladarHacia(new Vector3d(0, -100, -300)); //posición
return b; }
```

En el constructor inicializa la clase

```
public Principal() {
super();
inicializar();
applet = this; }
```

En el método inicializar que contiene el universo virtual se creará la escena agregando el *canvas* al *SimpleUniverse*

```
public void inicializar() {
setLayout(new BorderLayout());
GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
canvas3D = new Canvas3D(config);
canvas3D.setSize(1100, 800);
add("Center", canvas3D);
// canvas
simpleU = new SimpleUniverse(canvas3D);
    BranchGroup scene = createSceneGraph(simpleU);
```

```

scene.compile(); //Compila laescena
simpleU.getViewingPlatform().setNominalViewingTransform();
simpleU.addBranchGraph(scene);
scene2=scene;
    pickCanvas = newPickCanvas(canvas3D, scene);
    pickCanvas.setMode(PickInfo.PICK_BOUNDS);

```

Agrega al *canvas* la opción del mouse y me deja ver las coordenadas

```

canvas3D.addMouseMotionListener(this);
canvas3D.addMouseListener(this); } // fin

```

Prueba de Unidad

Las imágenes 3D editadas se añaden al Applet de Java 3D que posteriormente serán seleccionadas y eliminadas cada una de ellas con el fin de mostrar la siguiente capa, iniciando desde la capa piel luego la de músculos, la de órganos y la del esqueleto

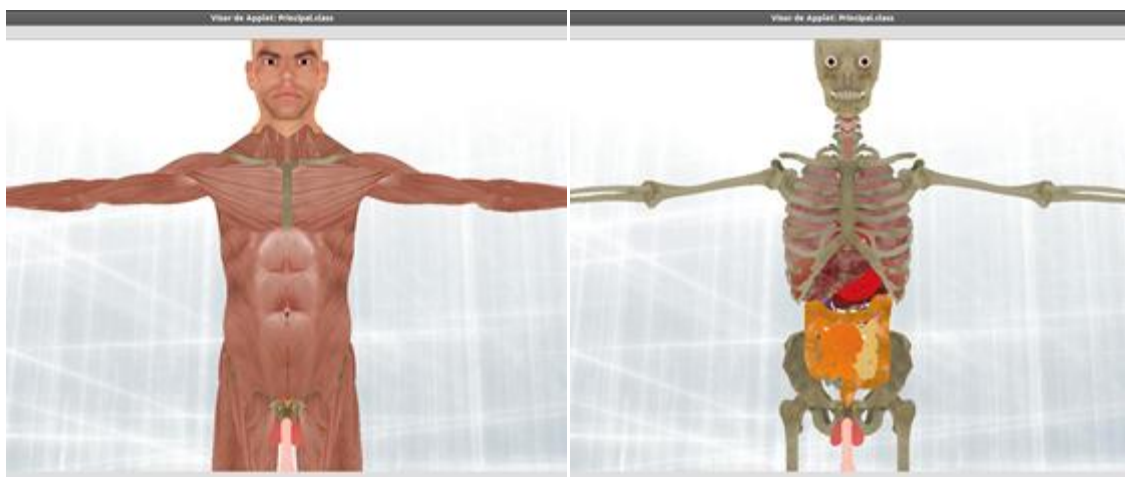


Figura 3-35 Ventana de Applet con cuerpo humano capa piel, capa músculos, capa órganos, capa sistema respiratorio y capa esqueleto

Fuente: Autores

Comprobación de errores

Error 01: *JavaHeadSpace* al cargar todos los objetos

El tamaño de las imágenes (texturas) en los objetos es muy grande por lo que al compilar genera el error de “*JavaHeadSpace*”, por lo que se debe aumentar el tamaño en memoria física o cambiar el proceso de 32 bits a 64 bits.

Solución de errores

Solución: Error 01

Se procede a cambiar la configuración de Eclipse para que se ejecute con un tamaño de 1024Mb ya que por defecto está predeterminado en 256Mb ver Anexo Compilar y cambiar argumentos en Eclipse tomando en cuenta que las imágenes son editadas en partes para reducir su tamaño.

HU3: Creación de una página Web para manipulación del Applet desarrollado

De acuerdo a la Tabla 3-7 Historia de Usuario HU3 previamente revisada se establecerá la estructura del sistema a ser utilizado durante el resto del proyecto para la página Web.

Iteración 1 Implementación de una página Web

El diseño y la implementación de la página Web sirve para que el Applet pueda ser visualizado desde un navegador Web por varios usuarios y con una interfaz agradable, para ello se usó el *tag*<APPLET> y sus atributos, con esto se puede acceder al Applet 3D y la página Web se la podrá visualizar desde cualquier parte de la red interna de la Universidad Politécnica Salesiana ya que se la colocará dentro del clúster.

El Applet desarrollado no debe tardar tanto en la carga, por tal motivo la página Web debe ser sencilla y poseer una interfaz simple.

Para cargar el Applet en HTML se requiere lo siguiente:

- Compilar el fichero y hacerlo “.class” mediante: javac
- Añadir los “.class” a un archivo de tipo “.jar”
- Escribir la página Web que contendrá al Applet y guardar el código con extensión .HTML
- Se requiere también un navegador preferible Google Chrome o Mozilla Firefox en su versión 3.6 o mayor.
- Añadir las librerías de Java 3D a la carpeta del navegador a usar.
- Para ejecutar el Applet se deberá lanzar un navegador y cargar la página HTML que contenga el Applet.

El código para el uso de un Applet dentro de un navegador es:

```
<HTML>
<BODY>
<APPLET code="Applet3D.class" width=400 height=400>
</APPLET>
</BODY>
```

Se requiere también un navegador preferible Google Chrome o Mozilla Firefox en su versión 3.6 o superior.

Añadir las librerías de Java 3D a la carpeta del navegador a usar.

En esta iteración se obtuvo la aceptación del cliente sobre los avances presentados por lo que se procede a su elaboración.

Diseño

Para el diseño de la página Web se sigue lo que la metodología ágil XP propone, tratar de evitar soluciones complejas y trabajar en una iteración a la vez.

Iteración 1. Diseño e implementación de una página Web

La página Web debe ser rápida, sencilla de usar, informativa y que contenga al Applet desarrollado.

Para la elaboración de la página Web se debe considerar algunos puntos:

- Tipo de página Web: informativa y educativa.
- Navegación apropiada: accesible y sencilla de usar.
- Contenido de la página Web: Applet 3D
- Aspectos estéticos: simple y acorde con los colores de la U.P.S.
- Aspectos técnicos: HTML, Java 3D (J3D).
- Multimedia en Internet: sin multimedia puesto que debe ser rápida.
- Texto: corto, comprensible e informativo.

La página Web a elaborar estará seccionada en tres partes: la primera dará la bienvenida al usuario al sistema, la segunda proporcionará información de lo que trata el aplicativo 3D así como de los paquetes que requiere instalar y la siguiente es la visualización del Applet 3D.

Puesto que se necesita una página Web rápida, sencilla y que no lleve mucho tiempo en crearla la estructura básica se la editará a conveniencia.



Figura 3-36 Plantilla para editar

Fuente: Autores

Codificación

Iteración 1. Diseño e implementación de una página Web

Plantilla HTML



Figura 3-37 Plantilla HTML

Fuente: Autores

Módulo: Edición de página Web

Proceso: Elaboración de Plantilla HTML

Descripción: La edición de la página Web se realizará a partir de una estructura sencilla como se indicó anteriormente.

En primera instancia se editará solo el archivo “index.html” que servirá de base para las otras dos partes. Se puede usar cualquier editor de texto sencillo, en este caso “Gedit” para luego ir agregando texto e imágenes como muestra la Figura 3-38.



Figura 3-38 Plantilla elaborada

Fuente: Autores

Luego de haber diseñado la plantilla con una interfaz amigable al usuario se copiará el código del archivo editado “index.html” para la edición de las otras páginas antes mencionadas. En el menú de la cabecera cambiar el “ID” a cada página para que se activen cada una cuando sean seleccionadas:

```
<li class="nav2" id="active"><a href="Informacion.html">Información</a></li>
<li class="nav3"><a href="Humano3D.html">humano3d</a></li>
```

```
<li class="nav2"><a href="Informacion.html">Información</a></li>
<li class="nav3" id="active"><a href="humano3d.html">Humano3D</a></li>
```


Por último editar la página “informacion.html” agregando datos del uso y configuración del Applet3D. Y agregar el Applet a la página “humano3d.html” dentro del cuerpo (body) de la página:

```
<!--CuerpoHumano3D -->
<h3>Cuerpo Humano<span> 3D</span>!</h3>
<applet code="Principal.class" archive="humano3d.jar" align="middle" height="200"
width="200">
<!-- /CuerpoHumano3D -->
```



Figura 3-39 Página “humano3d.html” con el Applet3D incluido

Fuente: Autores

Para cargar el Applet3D se debe configurar el navegador con las librerías de J3D para su correcta visualización (Ver Anexo Configuración Java 3D - API 1.4.0_01).

Pruebas de Unidad

En la página humano 3D el usuario visualiza el Applet 3D cuyas imágenes incorporadas son leídas desde el servidor Web.

A. Página Web de Inicio

La página de inicio de la aplicación es la de presentación, cuenta con algunas imágenes de la Universidad Politécnica Salesiana y un mensaje de bienvenida para los usuarios.

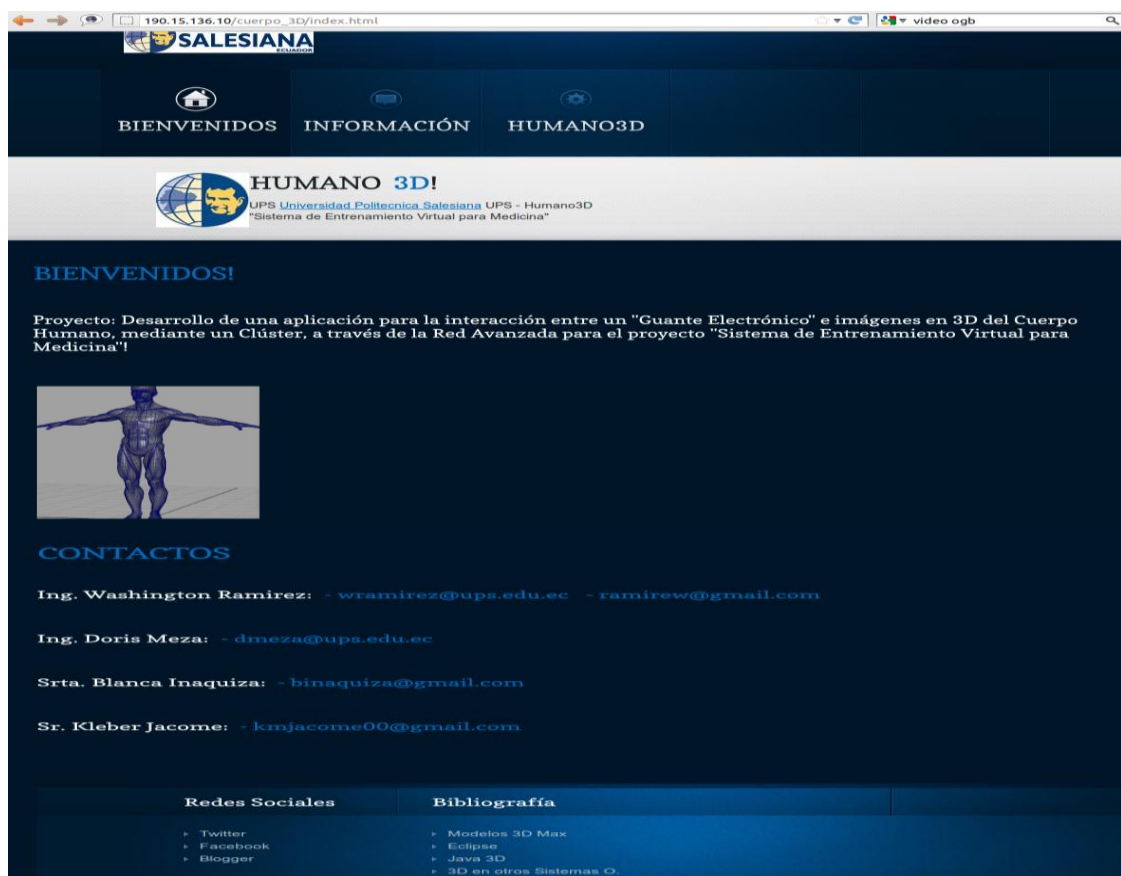


Figura 3-40 Página de inicio

Fuente: Autores

B. Página Web de Información

La página de información muestra una breve descripción de los requerimientos que el Applet necesita para poder ser visualizado en el navegador Web como indica la siguiente figura.



Figura 3-41 Página de información

Fuente: Autores

C. Página Web del Applet 3D

En esta página el usuario puede visualizar el Applet 3D, requiere un tiempo aproximado de cinco minutos, porque en las imágenes están incorporadas texturas de alta calidad y además son leídas desde el servidor Web.



Figura 3-42 Página de visualización del Applet 3D

Fuente: Autores

Comprobación de errores

Error 01: No cargan las imágenes 3D ni el diseño de la página

El diseño CSS y las imágenes no se muestran, tan solo el texto y los links.

Error 02: La página tarda mucho tiempo en cargar el Applet.

La página Web se demora y cada vez que se la actualiza se tarda mayor tiempo, especialmente la parte del Applet3D.

Solución de errores

Solución 01: No cargan las imágenes 3D ni el diseño de la página

No se cargan las imágenes debido a que los permisos que tienen los archivos de la Web no están habilitados. Cada vez que se agregan nuevas carpetas o imágenes hay que habilitar los permisos de ejecución donde se halla alojado la página Web, para el efecto se usa el siguiente comando:

```
$ chmod -R 777 /carpetaHTML
```

Solución 02: La página tarda mucho tiempo en cargar el Applet.

Se utiliza mayor tiempo al cargar el Applet debido a que este se almacena en “caché” para solucionar dicho problema se debe liberar la memoria cache al actualizar la página Web:

```
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE">  
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

Además se hace uso de un parámetro especial al momento de cargar el Applet, para que se ejecute con un argumento predeterminado de “*Java Head Space*” en 1024Mb, similar al usado en Eclipse:

```
<PARAM name="java_arguments" value="-Xmx1024m">
```

Si el proceso indicado se ejecutó de manera correcta, el tiempo de carga del Applet no debe extender los dos minutos.

Se ha terminado la iteración HU3, cumpliendo con la tarea de la historia de usuario.

HU4: Creación del software que gestione los datos provenientes del dispositivo electrónico y envíe dicha información a los aplicativos 3D

De acuerdo a la Tabla 3-8 Historia de Usuario HU4 previamente revisada se establecerá la estructura del sistema a ser utilizado durante el resto del proyecto para la interacción del dispositivo electrónico.

Iteración 1. Diseño e implementación de un aplicativo que permita interactuar entre el dispositivo electrónico y el aplicativo para manipular las imágenes 3D

Para elaborar el aplicativo que se comunique con el dispositivo electrónico se utilizará C++ por lo cual se requiere la instalación de aplicaciones y librerías como:

El JRE (Anexo Configuración de JRE) y JDK (Anexo Configuración de JDK) además de otras librerías como: mingw32, GLUT, X11, xlib, ruby, Motof, xtst (ver Anexo Librerías y paquetes de instalación C++).

Instalarla librería *libserial* para el manejo del puerto serial y de esta manera realizar la comunicación con el dispositivo electrónico (ver Anexo Librerías y paquetes de instalación C++)

Es necesario conocer en qué puerto está conectado el dispositivo electrónico para de esta manera poder realizar la manipulación caso contrario al realizar la ejecución del aplicativo este no compilará, para ello colocar el comando “*lsusb*” (lista los dispositivos USB y puertos USB existentes)

```
$ sudo lsusb
```

Para saber la lista de los dispositivos reconocidos y conectados colocar el comando “*dmesg*” ()

```
$ sudo dmesg
```

También se puede reconocer solo para el puerto USB con el siguiente comando:

```
$ sudo dmesg | grep usb
```

Para que el aplicativo ejecute y reconozca al dispositivo electrónico por el puerto USB se debe dar permiso cada vez que sea introducido con el siguiente comando en el FrontEnd:

```
#chmod a+rw /dev/ttyUSB0
```

En esta iteración se obtuvo la aceptación del cliente sobre los avances presentados por lo que se procede a su elaboración.

Diseño

Para el diseño del aplicativo que gestione los datos provenientes del dispositivo electrónico se siguió lo que la metodología ágil XP recomienda, trabajar en una iteración a la vez y evitar las soluciones complejas.

Iteración 1. Diseño e implementación de un aplicativo que permita interactuar entre el dispositivo electrónico y los aplicativos para manipular las imágenes 3D

Estructura del algoritmo dispositivo electrónico

Se requiere un programa que lea desde el puerto Serial los datos provenientes del dispositivo inalámbrico, conectado al USB. Controle, gestione esta información y envíe los datos haciendo que se generen pulsos de “*Key_Down*” en el teclado y a su vez realizando movimientos en el puntero del mouse (ratón):

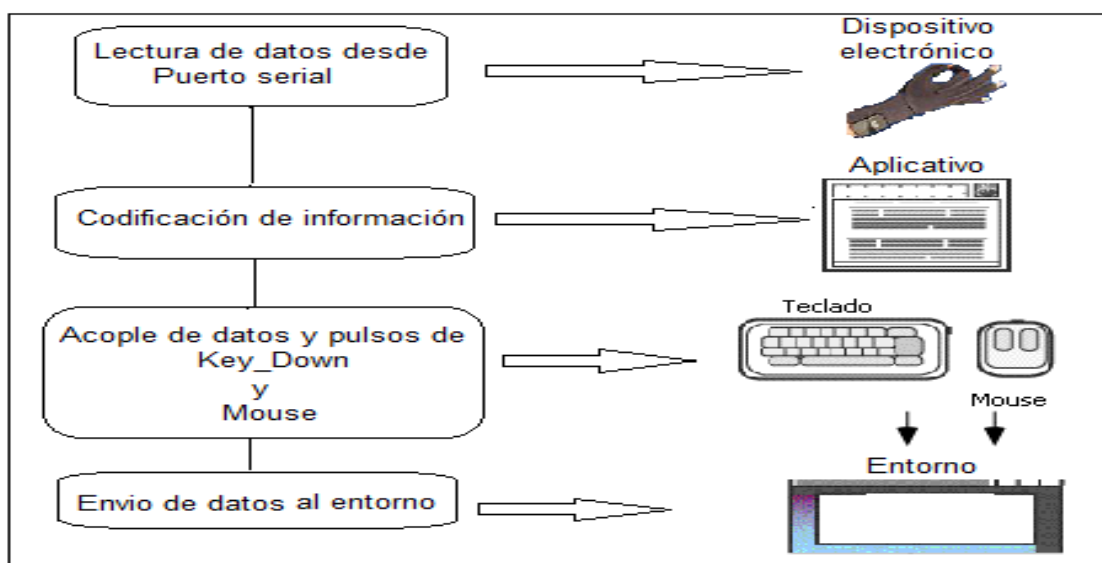


Figura 3-43 Fases para programación interacción Aplicativo – Dispositivo electrónico

Fuente: Autores

A. Diagrama de clases

Se ha estructurado el diseño de la aplicación en diferentes clases que se realizarán para su desarrollo:

Diagrama de clases para el aplicativo Guante Electrónico		
Clase	Función	Diagrama
Main	Es la clase principal que ejecuta el aplicativo. Incluye algoritmos de inicialización y llamada a las clases necesarias para activar el puerto.	<pre> main app : XtAppContext label : XmString al : Arg ac : Integer button2 : Widget button4 : Widget form : Widget container : Widget toplevel : Widget controlPort : bool w : Widget client_data : XtPointer event_data : XtPointer argc : Integer argv : Char iniciaPuerto(w,client_data,event_data) cierraPuerto() imprime_texto(w,client_data,event_data) cierra_programa(w,client_data,event_data) main(argc,*argv) </pre>

Diagrama de clases para el aplicativo Guante Electrónico		
Clase	Función	Diagrama
main.h	Encargada de inicializar las variables locales y globales a usar en la clase main.	<pre> classDiagram class main_h { mouse_x : Integer mouse_y : Integer cierraPuerto : Void } </pre>
puerto	Es la clase encomendada en configurar, activar, iniciar y leer el puerto USB.	<pre> classDiagram class puerto { nextByte : Char x : Integer y : Integer boton : Integer namePort : String cmd : String strx : String stry : String puerto(namePort) controlPantalla(x,y,boton) readByte() puerto() } </pre>
puerto.h	Inicializa las variables locales, globales y las funciones utilizadas en la clase puerto.	<pre> classDiagram class puerto_h { puerto : Puerto readByte : Void str_to_int : Integer controlPantalla : Void flag : bool x : Integer y : Integer z : Integer i : Integer boton : Integer data : String buff : String } </pre>

Diagrama de clases para el aplicativo Guante Electrónico		
Clase	Función	Diagrama
periféricos	Es la clase que realiza un control de los datos leídos desde el puerto USB, esta información es analizada y controlada para activar ya sea el movimiento, click del mouse o el pulso de teclado.	<pre> classDiagram class periféricos { contador1 : Integer contador2 : Integer bandera1 : bool bandera2 : bool button : Integer display : Display event : XEvent x : Integer y : Integer controlIT : Integer b : Integer ControlID() ControlMouse() click(button) move(display,x,y) teclashabiles(controlIT) inicioControl(x,y,b) periféricos() } </pre>
periféricos	Inicializa las variables y funciones para que la clase periféricos las utilice.	<pre> classDiagram class periféricos.h { inicioControl : Integer teclashabiles : Void comparaVariacion : Void mousex : Integer mousey : Integer newOperation() } </pre>

El Diagrama de clases para el aplicativo con el guante electrónico de datos (ver Figura 3-27) se diseña con la librería serial, donde la clase Main va a contener las clases necesarias para activar el puerto (ver anexo 5.4.1.2).

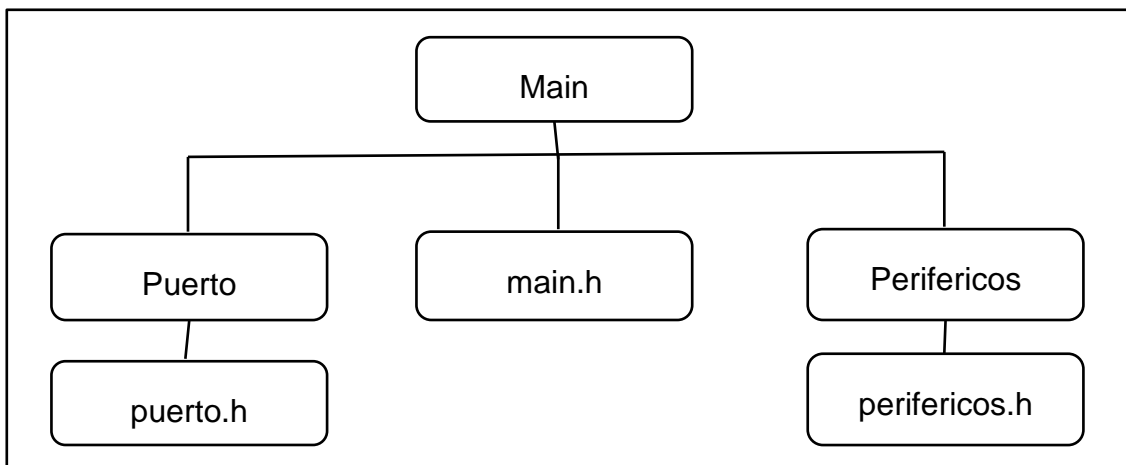


Figura 3-44 Diseño de clases guante electrónico de datos

Fuente: Autores

B. Descripción de librerías

Para iniciar la programación de C++ en este aplicativo se debe llamar a una serie de librerías las cuales se clasifican en 4 tipos:

- Librerías básicas para el uso de C++

Librerías Básicas	
Librería	Descripción
#include <stdio.h>	Funciones de Entrada/Salida estándar.
#include <string.h>	Manejo de cadenas de caracteres.
#include <stdlib.h>	Es el archivo de cabecera de la biblioteca estándar de propósito general.
#include <math.h>	Funciones matemáticas.
#include <unistd.h>	proporciona acceso con cabeceras propias de Unix, similar a <i>stdio.h</i>
#include <iostream>	contiene información de Entrada/Salida, con mejor función y más fácil de manejar que <i>stdio.h</i>

Tabla 3-18 Librerías C++

Fuente: Autores

- Librerías gráficas y de uso de periféricos X11

Librerías gráficas y de periféricos	
Librería	Descripción
#include <X11/Xlib.h>	Se la usa para llamar al servidor X y crear ventanas.
#include <X11/Xresource.h>	Declara funciones, tipos y símbolos para el trabajo con recursos.
#include <X11/Xutil.h>	funciones, tipos y símbolos para comunicación entre clientes
#include <X11/extensions/shape.h>	Proporciona funciones gráficas.
#include <X11/extensions/XTest.h>	Declara funciones para el uso de periféricos, mouse y teclado.

Tabla 3-19 Librerías Gráficas

Fuente: Autores

- Librería para el uso del Puerto USB

Librería para el puerto USB	
Librería	Descripción
#include<SerialStream.h>	librería de manejo de Puerto Serial

Tabla 3-20 Librerías puerto USB

Fuente: Autores

- Librerías realizadas para un mejor manejo del aplicativo

Librería realizadas	
Librería	Descripción
#include "Puerto.h"	declara variables del puerto serial
#include "perifericos.h"	declara e inicializa variables de periféricos.cpp
#include "main.h"	inicializa variables de main.cpp

Tabla 3-21 Librerías propias

Fuente: Autores

C. Lectura de datos mediante la librería Libserial

Para acceder a la librería “*Libserial*” se usa el código “*SerialStream*”, ya que esta instancia permite la conexión con el puerto y para usarla se accede usando la siguiente cabecera:

```
#Include <SerialStream.h>
using namespace Libserial;
```

Una vez que ya se conoce a que puerto está conectado el dispositivo electrónico, para abrir el puerto USB e iniciar con la lectura de datos se agregará el siguiente comando:

```
Open("dev/ttyUSB0");
```

También se lo puede realizar de la forma siguiente:

```
Open(string nombrePuerto);
MySerialStreamserial_port("/dev/ttyUSB0");
```

D. Tipos de datos recibidos (entrada)

Los datos obtenidos desde el dispositivo USB son cadenas de caracteres las cuales son divididas en partes para su manipulación. El programa implementa una variable tipo *string* con el propósito de acumular los caracteres recibidos, estos datos a su vez son divididos para obtener sus valores correspondientes y luego transformarlo a enteros para ser usados en el programa:

```
string x = cad.substr(nx+1,ny-1);
```

E. Control de datos recibidos

El control se lo realizará mediante métodos con los cuales se podrán adaptar los datos de entrada al aplicativo, de manera que sean de fácil manipulación.

Codificación

Iteración 1. Diseño e implementación de un aplicativo que permita interactuar entre el dispositivo electrónico y los aplicativos para manipular las imágenes 3D

Clase: Puerto Serial

Módulo: Adquisición de datos

Proceso: Espera la activación de la conexión y si es correcta transmite datos desde el dispositivo electrónico.

Descripción:

Los algoritmos y métodos para la adquisición de datos se lo realiza en la clase “*puerto.cpp*” exclusiva para la configuración y uso de los parámetros de apertura del puerto USB emulado. El constructor de la clase es detallado a continuación:

Parte del código es descrito a continuación:

Se llama a la clase periféricos

Se inicia el constructor colocando las variables x, y, z igualando a cero y se inicializa una bandera en true=verdadero.

```
Periféricos perif;
Puerto::Puerto(stringnamePort) { // Constructor de “puerto.cpp”
this->flag = true;
this->Open(namePort); // Inicializado en “puerto.h”
```

Verifica si en el puerto USB hay conexión

```
if ( ! this->good()) {
    std::cerr<< "Error al abrir el puerto serial."<<std::endl ;
    exit(1) ;    }
```

Vuelve a restablecer la velocidad de *BaudRate* para el uso del Puerto Serial

```
this->SetBaudRate(StreamBuf::BAUD_57600); //
if ( ! this->good()) {
    std::cerr<< "Error: no se puede establecer el baudrate" <<std::endl ;
    exit(1) ;    }
```

Tamaño de caracteres ingresados al puerto

```
this->SetCharSize(StreamBuf::CHAR_SIZE_8);
if ( ! this->good()) {
    std::cerr<< "Error: no se puede establecer el tamaño del carácter" <<std::endl ;
    exit(1) ;
    }
```

Número de bits de parada en el puerto serial

```
this->SetNumOfStopBits(1);
if ( ! this->good()){
    std::cerr<< "Error: no se puede configurar los bits de parada."<<std::endl ;
    exit(1);
    }
```

Existencia de paridad en el puerto serial

```

this->SetParity(StreamBuf::PARITY_NONE);
if ( ! this->good()){
std::cerr<< "Error: no se puede deshabilitar la paridad ." <<std::endl ;
exit(1);
}

```

Control de flujo

```

this->SetFlowControl(StreamBuf::FLOW_CONTROL_NONE);
if ( ! this->good() ) {
std::cerr<< "Error: no se puede deshabilitar el control de flujo." <<std::endl;
exit(1) ;
}
} // Fin Constructor

```

Clase: Main

Módulo: Clase principal y gráfica

Proceso: Abre el puerto serial e inicializa el *frame* gráfico al cargar la clase principal (main.cpp)

Evento: Lectura de USB y visualización de Contenedor *XtVaCreateWidget*.

Descripción:

En la clase *main.cpp* (Clase principal) se tomará lectura de los datos ingresados por el dispositivo electrónico a través del puerto y se enlazarán mediante los parámetros de la clase puerto.cpp como se muestra a continuación:

```

Void MySerialStream::read_byte(boolparada)

```


Leídos y evaluados los datos entrantes al puerto se procesarán con el método “*data_processing*”:

```
Void MySerialStream::data_processing(string cad, unsigned int count)
```

El código de la clase *main.cpp* se detalla a continuación:

Inicializa el Puerto y llama a las variables de la clase “*puerto.h*”

```
#include "puerto.h" // librería para la configuración del puerto serial
#include "main.h" // librería de variables globales e inicializadas para la clase principal
void iniciaPuerto(Widget w, XtPointer client_data, XtPointer event_data) {
    Puerto serial_port("/dev/ttyUSB0");
    serial_port.clear();
    cout<<"Puerto Iniciado...\n"<<endl;
    controlPort=true;
    serial_port.readByte();
    usleep(100);
    serial_port.Close(); }
```

En el método principal se realiza la inicialización de datos y variables para gráficos y el *widget* se usa para la visualización de la ventana de Inicio

```

int main(intargc, char *argv[]){
Widget toplevel, form, button2, button4, container;
XtAppContext app;
XmString label;
Argal[10];
int ac;
toplevel = XtVaAppInitialize(&app, "Editor", NULL, 0, &argc, argv, NULL, NULL);
form = XtVaCreateManagedWidget("form", xmFormWidgetClass, toplevel, NULL);
ac = 0;
XtSetArg(al[ac], XmNeditMode, XmMULTI_LINE_EDIT); ac++;
XtSetArg(al[ac], XmNscrollVertical, TRUE); ac++;

```

Contenedor de la visualización (*Form*)

```

container = XtVaCreateManagedWidget("box", xmRowColumnWidgetClass, form,
XmNrightAttachment, XmATTACH_FORM,
XmNleftAttachment, XmATTACH_FORM,
XmNbottomAttachment, XmATTACH_FORM,
XmNorientation, XmHORIZONTAL,
NULL);

```

Clase: Periféricos

Módulo: Inicialización de periféricos

Proceso: Controla los datos entrantes y gestiona el uso de mouse y teclado.

Evento: *mouseclick*, *mousemove* y habilitación de teclas mediante *xtestfakekeyevent*.

Descripción:

Primero se debe controlar los datos ingresados y después inicializar los periféricos con los datos permitidos:

Parte del código es descrito a continuación:

Con el método de control de datos para movimiento uniforme se establece la velocidad por conteo de información e ingreso de datos, mediante banderas

```
int ControlD(){
    usleep(1);
    if(cont03>=15 &&cont03<=40){
        bandera1=0;
        if(bandera1==0){
            cont03=0;
            bandera1=1;
        }
        return 1; }
    usleep(1);
    }else {
        bandera1=0;
        return 0;}
    usleep(1);
    return 0; }
```

Para la manipulación de *Keyboard* se realiza el siguiente método

```
void Perifericos::teclashabiles(intcontrolIT){
    Display *xdpy;
    if ( (display_name = getenv("DISPLAY")) == (void *)NULL) {
        fprintf(stderr, "Error: DISPLAY no inicializado\n");
        exit(1); }
```

```

if ( (xdpy = XOpenDisplay(display_name)) == NULL) {
    fprintf(stderr, "Error: Display NO Abierto: %s", display_name);
    exit(1); }

```

Inicialización de teclas utilizadas –*keyboard*

```

int tecla_S, tecla_C, tecla_V, tecla_F_Up, tecla_F_Down, tecla_F_Iz, tecla_F_De,
tecla_O, tecla_D;

tecla_Q = XKeysymToKeycode(xdpy, XStringToKeysym("Q"));
tecla_E = XKeysymToKeycode(xdpy, XStringToKeysym("E"));
tecla_A = XKeysymToKeycode(xdpy, XStringToKeysym("A"));
tecla_D = XKeysymToKeycode(xdpy, XStringToKeysym("D"));
tecla_O = XKeysymToKeycode(xdpy, XStringToKeysym("O"));
tecla_F_Up = XKeysymToKeycode(xdpy, XStringToKeysym("Up"));
tecla_F_Down = XKeysymToKeycode(xdpy, XStringToKeysym("Down"));
tecla_F_Iz = XKeysymToKeycode(xdpy, XStringToKeysym("Left"));
tecla_F_De = XKeysymToKeycode(xdpy, XStringToKeysym("Right"));

```

Teclas habilitadas, iniciar *keypress*

```

switch (controlT) {
    case 3:
        XTestFakeKeyEvent(xdpy, tecla_E, True, CurrentTime);
        ...
        break;
    case 4:
        XTestFakeKeyEvent(xdpy, tecla_D, True, CurrentTime);
        ...
        break;
    ...
}

```

```

case 8:
    cout<<"Puerto Finalizado!\n"<<endl;
    exit(1); // Salir de la aplicación
    break;
    ...
default: cout<<"Error tecla No definida!!! "<<endl;

```

Visualiza físicamente el evento y cierra el display

```

XFlush(xdpy);
XCloseDisplay(xdpy);
usleep(10);}

```

Librerías para inicialización de variables

Módulo: Inicialización de variables

Proceso: Carga las variables al ser llamadas ya sean de tipo local o global.

Descripción:

Controlar los datos ingresados, después inicializar los periféricos con los datos permitidos.

Para el manejo del Puerto Serial se utilizará *using* para librerías

En la clase *serialstream* se crea variables públicas, privadas y para almacenar las medidas de los sensores para el manejo del puerto serial

```

using namespace LibSerial;
using namespace std;
class Puerto:public SerialStream{
public:
    Puerto(string namePort);
    Puerto();
    void readByte();
    int str_to_int(string cad);
    void controlPantalla(int x, int y, int boton);
private:
    bool flag;
    ... };

```

Pruebas del dispositivo electrónico

Los datos ingresados desde el dispositivo electrónico son cadenas de caracteres (*String*) que dan valores para el movimiento del mouse en X e Y y un dato que varía entre 0 y 8 el cual es asociado para generar eventos propios de un mouse, además se puede anexar funciones del teclado, para manipular los objetos 3D. Se cuenta con una configuración predeterminada y se la puede cambiar desde el código realizado en C++, a continuación se detallan las que se utilizarán:

Para mejores resultados se procede a configurar cada uno de los datos ingresados, dependiendo de su función se establece en el dispositivo electrónico la posición más apropiada para cada acción y así el usuario interactúe con la aplicación, en la Figura 3-45 indica como cada dato va a ir asociado a un evento.

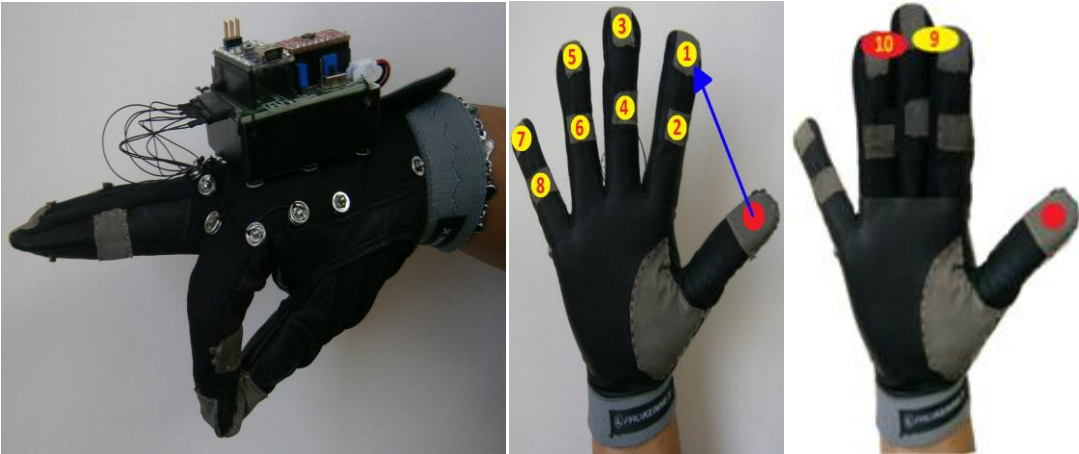








Figura 3-45 Eventos asociados al dispositivo electrónico




Fuente: Autores

La Tabla 3-22 muestra la acción que efectuará el dispositivo electrónico cuando cada dato sea ingresado:

Acciones a realizar con el dispositivo electrónico, teclado y mouse				
#	Evento	Dato de entrada	Posición en el dispositivo electrónico	Configuración asociada
1	--	"9"		Inicia datos de entrada al presionar el pulgar contra la punta del índice y la punta del dedo medio.

Acciones a realizar con el dispositivo electrónico, teclado y mouse				
#	Evento	Dato de entrada	Posición en el dispositivo electrónico	Configuración asociada
2	--	"10"		Paro momentáneo de datos al presionar el pulgar contra la punta el dedo medio y la punta del dedo anular.
3	Mouse Button1 (Applet) Supr (aplicativo con OSG)	- "1"		Retira partes del cuerpo al clic al presionar el pulgar contra la punta el dedo índice.
4	KEY - "A"	"2"		Giro Anti-horario al presionar el pulgar contra el medio el dedo indice.
5	KEY - "D"	--		giro Horario

Acciones a realizar con el dispositivo electrónico, teclado y mouse				
#	Evento	Dato de entrada	Posición en el dispositivo electrónico	Configuración asociada
6	KEY - "E"	"3"		Zoom de aumento de tamaño al presionar el pulgar contra la punta el dedo medio.
7	KEY - "Q"	"4"		Zoom de disminución de tamaño al presionar el pulgar contra el medio del dedo medio.
8	KEY - "←"	"5"		Desplaza a la izquierda al presionar el pulgar contra la punta el dedo anular.

Acciones a realizar con el dispositivo electrónico, teclado y mouse				
#	Evento	Dato de entrada	Posición en el dispositivo electrónico	Configuración asociada
9	KEY - "→"	"6"		Desplaza a la derecha al presionar el pulgar contra el medio del dedo anular.
10	KEY - "↑"	"7"		Desplaza hacia arriba al presionar el pulgar contra la punta el dedo meñique.
11	KEY - "↓"	--	--	Desplazar hacia abajo
12	--	"8"		Fin de la trama de datos al presionar el pulgar contra la punta el medio del dedo meñique.
13	KEY - "P"	--	--	Volver posición original

Acciones a realizar con el dispositivo electrónico, teclado y mouse				
#	Evento	Dato de entrada	Posición en el dispositivo electrónico	Configuración asociada
14	KEY - "H" (aplicativo con OSG)	--	--	Guarda el objeto seleccionado
15	KEY - "G" (aplicativo con OSG)	--	--	Guarda toda la imagen 3D
16	Mouse X	entre "1" a "1024"	--	Posición puntero X
17	Mouse Y	entre "1" a "1024"	--	Posición puntero Y

Tabla 3-22 Configuración asociada a los eventos del mouse

Fuente: Autores

Configuraciones de resultados

De las pruebas efectuadas se pudo determinar cuáles son los datos apropiados para interactuar con la aplicación de tal manera que sean fáciles de realizar. Para una mejor interacción con la aplicación se utiliza el guante derecho, aprovechando de esta forma los gestos más sencillos de utilizar, y a su vez que el usuario pueda recordar.

El dispositivo electrónico es activado por el software elaborado tomando control del puntero del mouse en el aplicativo 3D, con lo cual se mapea las diferentes acciones como la selección, traslación, zoom, rotación y eliminación de objetos con un gesto, es decir con el movimiento del dispositivo electrónico para poder sensor los valores enviados de los sensores de contacto y de movimiento.

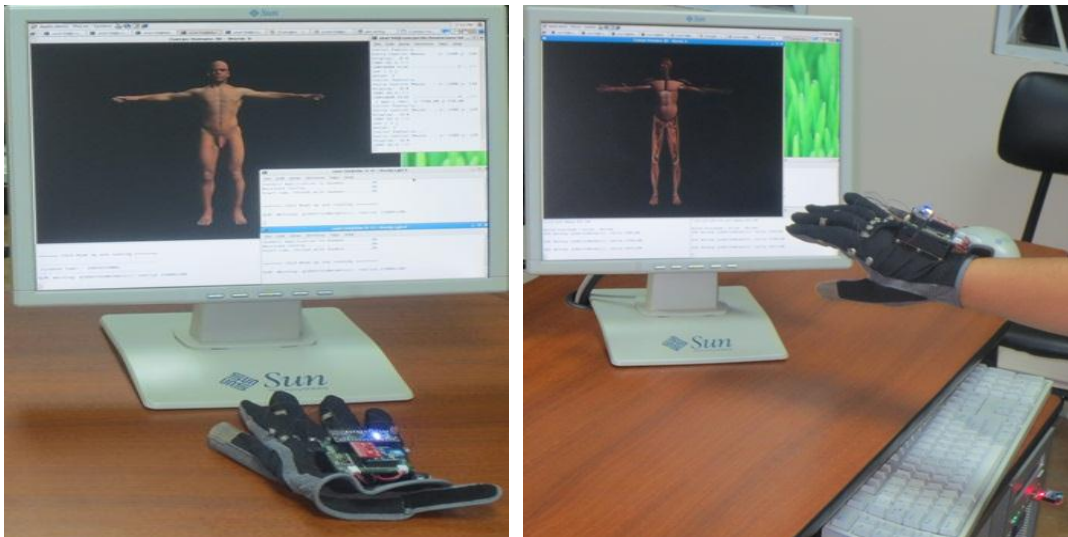


Figura 3-46 Dispositivo electrónico activado y controlando el aplicativo

Fuente: Autores

Comprobación de errores

Error 01: No reconoce el puerto USB

Al conectar el dispositivo electrónico y ejecutar la aplicación envía un mensaje de no reconocer el puerto USB.

Solución de errores

Solución: Error 01

Cada vez que es conectado el puerto USB se debe volver a dar permisos como se indicó anteriormente en la iteración 1 de esta historia de usuario.

Se ha terminado la iteración HU4, cumpliendo con la tarea de la historia de usuario respectivo.

HU5: Aplicativo desarrollado incluido en el Clúster

De acuerdo a la Tabla 3-9 Historia de Usuario HU5 previamente revisada se establecerá la estructura del sistema a ser utilizado durante el resto del proyecto para incluir el aplicativo 3D en el Clúster.

Iteración 1. Configuración de herramientas para el uso y paralelización de la aplicación 3D

Uno de los requerimientos del proyecto es que las imágenes deben estar soportadas sobre un Clúster y paralelizadas en los nodos.

Los requerimientos necesarios para instalar Rocks son:

- Un conjunto de máquinas de arquitectura similar (nodes), cada una con una interfaz de red, disco duro con capacidad para más de 7 GB y memoria RAM superior a 256 MB.
- Un data *switch* (o varios) con un número de puertos mayor al doble del número de máquinas disponibles para darle escalabilidad.
- Una máquina con 2 interfaces de red, capacidad en disco duro igual o superior a 20 GB, y memoria RAM superior o igual a 512 MB FrontEnd.
- Cables de red en número y longitud suficiente.
- Mueble o Rack con espacio apropiado para los chasis de las máquinas y eventualmente para el FrontEnd, con acceso apropiado a la parte de atrás de los equipos.
- Una habitación con ventilación o refrigeración adecuada para los niveles de disipación de calor de todos los equipos combinados.
- Adicionalmente se requiere una fuente de suministro eléctrico o 'UPS', para alimentar al menos una máquina por más de 10 minutos.
- Para la instalación del Sistema Operativo se debe disponer de los siguientes medios (*rolls*), se lo puede descargar del sitio de Rocks Clúster[75]

a) Kernel Roll

- b) Core Roll
- c) OS Roll
- d) Base Roll
- e) Ganglia Roll
- f) Web-Server Roll

Cualquier otro *roll* que considere necesario (Cóndor, Bio, Viz.HiperWorks) se lo puede instalar junto con el S.O.

Para continuar con la configuración del “FrontEnd”, antes de la instalación de los nodos de cómputo completar lo siguiente:

- Creación de cuentas de usuario
- Preparación del */export/apps*
- Instalación de software desde *tarballs*
- Instalación de paquetes desde *RPM's*
- Preparación y creación de la distribución

A. Comandos útiles para trabajar con el Clúster

El Clúster permite realizar tareas como: agregar y eliminar usuarios, luego de la creación de usuarios se puede comprobar que existe un directorio llamado “/home/usuario” accesible desde “/export/home/usuario” y desde allí se exporta los archivos a ser ejecutados en los nodos secundarios.

A continuación se describen algunos comandos que se usarán para trabajar y monitorear los procesos en el Clúster y los nodos.

Para reiniciar un nodo esclavo colocar en un terminal de FrontEnd

```
#rocks run host compute-0-0 reboot
```

Para apagar un nodo esclavo colocar en un terminal de FrontEnd

```
#rocks run host compute-0-0 poweroff
```

Para ingresar al nodo tipear:

```
#sshroot@compute-0-0
```

Para eliminar un nodo esclavo colocar en un terminal de FrontEnd

```
#insert-ethers --remove compute-0-0;  
#rocks list host;  
#rocks sync config
```

Cada vez que el nodo se apague se tiene que mandar a sincronizar los usuarios para de esta manera poder ingresar a los nodos.

```
#rocks sync users
```

B. Monitoreo de procesos:

Para ver los procesos que están ejecutándose se puede hacer uso del siguiente comando:

```
$ ps aux
```

Para mostrar todos los procesos en ejecución tipear

```
$ ps -u <nombre-de-usuario>
```

Para mostrar los procesos en ejecución de un determinado usuario.

```
$ ps aux | grep <filtro>
```

Para ver la carga a la que está sometido el nodo se ejecuta el comando *uptime*. Con este comando se observa el número de usuarios conectados en ese momento y la carga, donde el primer campo indica el promedio de carga en los últimos 5 minutos, el segundo los últimos 10 minutos y el tercero durante los últimos 15 minutos, esto quiere decir que cuanto más se acerque el valor del promedio al número de procesadores del sistema; mayor será la ocupación del mismo y si superase el número de procesadores significaría que está muy cargado el sistema y no se deberían de mandar más trabajos hasta que se descargue un poco.

Para el monitoreo de recursos por la Web se inicia identificando al nodo maestro con su dirección IP y a través de la interfaz de Ganglia que viene instalada por defecto en Rocks se consigue visualizar en los nodos la carga que realiza cuando se manda a generar un evento en el aplicativo.

Para esto en el navegador se debe introducir la dirección IP o DNS del clúster, conjuntamente con la palabra Ganglia al final (<http://190.15.136.20/ganglia>).

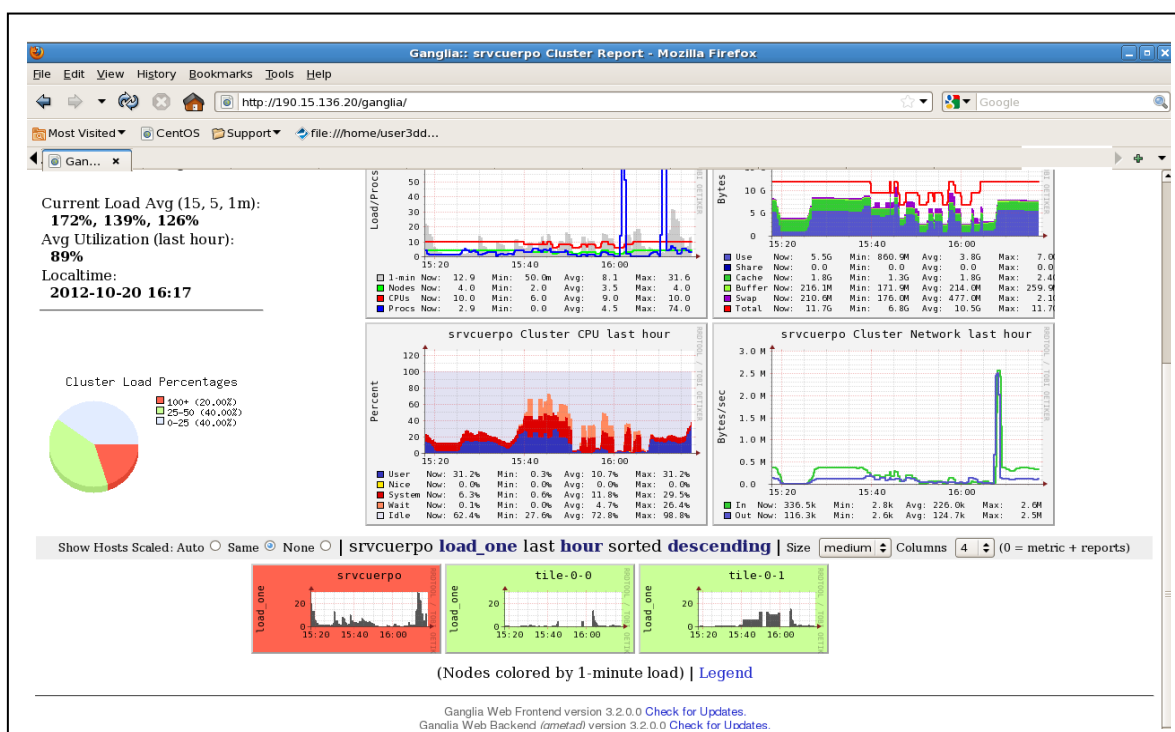


Figura 3-47 Monitoreo del Clúster con Ganglia

Fuente: Autores

C. Configuración de Virtual Host

Para que sea posible la visualización de la página Web desde cualquier parte interna de la red de la Universidad Politécnica Salesiana, se colocará a la misma dentro de un Clúster; el cual contendrá el servidor Web.

Se creará un “Virtual Host “en Apache donde se colocará “Virtual host *:80” para indicar que puede atender cualquier IP por el puerto 80, para lo cual se abre una solapa en consola y se tipea el siguiente comando como FrontEnd para abrir el archivo:

```
# nano /etc/httpd/conf/httpd.conf
```

Como primera instancia se realizará una copia del archivo “*httpd.conf*” para tener un respaldo por si se daña la configuración de inicio del sistema, con lo cual se procederá posteriormente a editarlo.

Diseño

Para el diseño de la aplicación se siguió la recomendación de la metodología ágil XP, se trató en lo posible de evitar las soluciones complejas y se trabajó en una iteración a la vez.

Iteración 1. Configuración de Virtual Host

Para realizar la configuración en el “Virtual Host “en Apache se colocará en el archivo “*httpd.conf*” en cada campo los datos que especifican la ubicación del archivo que contiene el aplicativo donde:

- **ServerAdmin:** Es la dirección email del administrador (190.15.136.10)
- **DocumentRoot:** Es la ruta donde están ubicados los archivos del sitio Web asociados a ese “Virtual Host” (/export/home/user3d/wwwroot)
- **ErrorLog:** Es el archivo .log donde se almacenará los errores que se generen en el sitio Web.

- **CustomLog:** Es el archivo .log donde se almacenará los accesos a este sitio Web.

La Figura 3-48 muestran las líneas de ejemplo que serán editadas luego colocando los datos respectivos.

```
<VirtualHost *:80>
    ServerAdmin webmaster@sitioa.com
    DocumentRoot "C/Servidor/Web/SitioA"
    ServerNamesitio
    ErrorLog logs/SitioA-error_log
```

Figura 3-48 VirtualHost en Apache[76]

Fuente: Ángel Acaymo M. G., Servidor Web Apache, PHP, MySQL., 2009

Codificación

Iteración 1. Configuración de herramientas para el uso y paralelización de la aplicación 3D

Módulo: Configuración del Clúster Rocks

Proceso: Instalación y configuración de Rocks Clúster Viper 5.4

Descripción:

Para proceder con la instalación de Rocks Clúster Viper 5.4 se deben introducir los siguientes datos cuando sean requeridos como muestra la Figura 3-49:

- 1) **Descargar el ISO Rocks Clúster Viper 5.4 para 64 bits y los rolls que se requieran instalar, como el VIZ ROLL y el HIPerWorks ROLL[75]**
- 2) **Instalar el Rocks Clúster con los siguientes datos:**
 - a) Red 1. (Red comercial) Eth0:192.168.1.100 – 255.255.255.0
 - b) Red 2. (Red Avanzada 2.0) Eth1:190.15.136.20 – 255.255.255.224

- c) Gateway: 190.15.136.1
- d) DNS servers: 8.8.8.8
- e) Usuario: srv cuerpo.ups.edu.ec
- f) Time Configuration: América/Guayaquil
- g) NTP Server: Inocar.ntp.ec

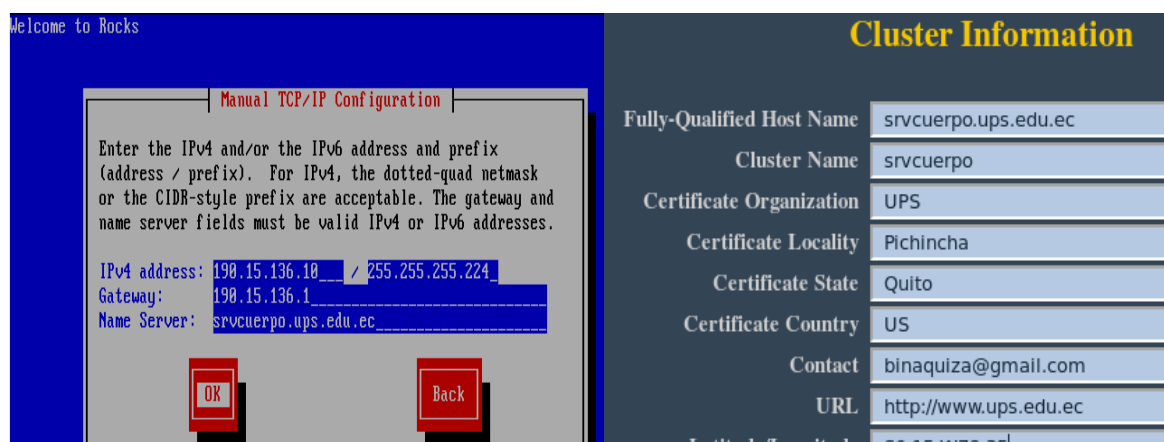


Figura 3-49 Instalación de Rocks Clúster

Fuente: Autores

Una vez instalado completamente el “FrontEnd” como indica la Figura 3-49 se procede a ajustar detalles de la configuración de la máquina y la preparación de la distribución, la cual será ubicada en los nodos.

3) Administración de Usuarios

Para agregar un “usuario” se debe crear su cuenta.

Creación básica de una cuenta:

```
#useradd user3d
```

Asignación correcta del home del usuario:

```
#usermod -d /export/home/user3d user3d
```

Asignación de la contraseña:

```
#passwd user3d
```

Sincronización de los archivos de usuario:

```
#rocks sync users
```

Para eliminar un usuario si se requiere ver anexo Eliminación de cuentas de usuario.

4) Instalar un nodo hijo, como FrontEnd colocar:

```
#insert-ethers
#rocks list host
#rocks sync config
```

Sincronizar usuarios, configuraciones, computadores, dispositivos con:

```
#rocks sync device
#rocks sync config
#rocks sync hosts
#rocks sync users
#rocks sync tile
```

Iteración 2 Configuración de Virtual Host

Módulo: Configuración de Virtual Host

Proceso: Creación de un servidor virtual

Descripción:

Dentro del archivo se debe buscar la “*NameVirtualHost*” y quitar las líneas de comentario que tiene al inicio de línea o escribirla si es necesario:

```
NameVirtualHost *:80
```

Colocar la ruta a la que se quiere direccionar:

```
<VirtualHost *:80>
DocumentRoot /export/home/user3d/wwwroot
ServerName 190.15.136.10
</VirtualHost>
```

Para guardar la configuración y salir del archivo colocar en consola lo siguiente:

```
Crtl+o y "Enter"
```

Para que restauren los valores que se han cambiado se realiza lo siguiente en consola:

```
/etc/init.d/httpdrestart
```

También se puede usar para restaurar los valores paso a paso el siguiente código:

```
/etc/init.d/httpd stop
/etc/init.d/httpd start
```

Iteración 3 Distribución de datos

Ubicación de archivos

El archivo que contiene las imágenes .JPG, los .OBJ del cuerpo humano y las páginas HTML, deberán ser colocados dentro de la carpeta del servidor HTML para ser accedidos desde la red interna de la Universidad Politécnica Salesiana, a los cuales se les debe proporcionar permisos de usuarios para accederlos.

```
chmod -R 777 cuerpo_3D
```

Pruebas con el Applet 3D

Las pruebas realizadas mostraron que el Applet puede ser visualizado desde cualquier parte de la Universidad Politécnica Salesiana

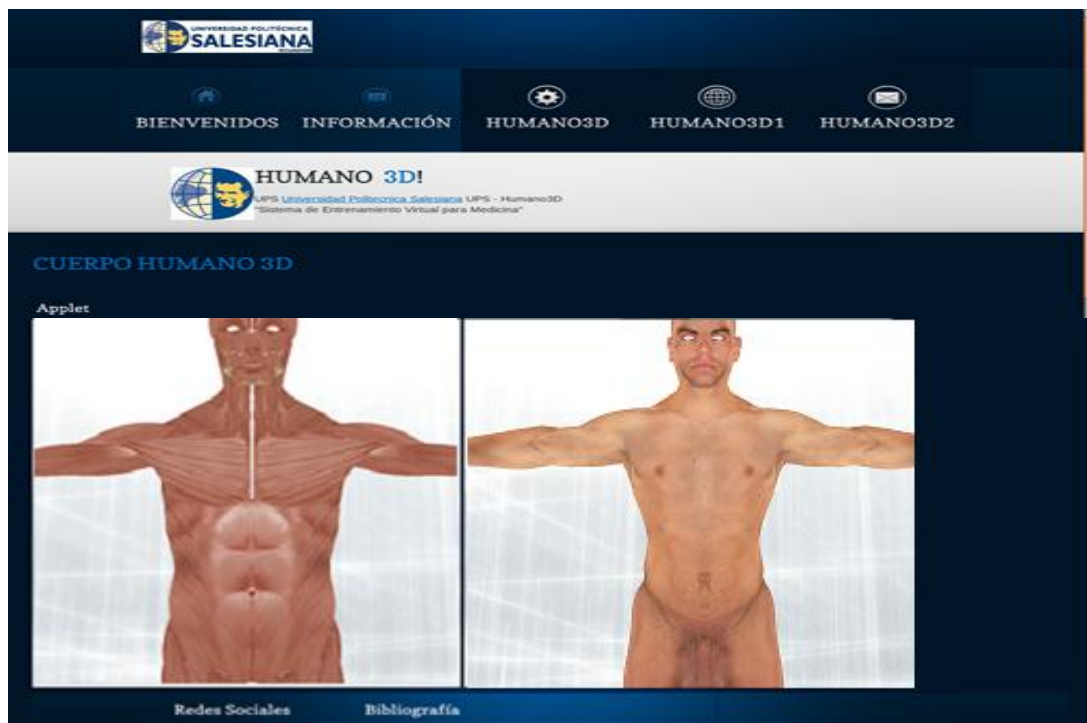


Figura 3-50 Applet incluido dentro del Clúster

Fuente: Autores

Comprobación de errores

Error 01: La página Web ejecuta el Applet en una ventana de tamaño 100x100 pixeles.

Al incluir el Applet en la página Web esta sólo se ejecuta en un tamaño de ventana de 100x100 pixeles y al momento de aumentar la ventana a más de 100x100 esta se visualiza, pero al intentar seleccionar y quitar el primer objeto del Applet 3D la ventana se torna gris como pretendiendo ejecutar la acción más no vuelve a cargar el Applet.

Solución de errores

Solución 01: La página Web ejecuta el Applet en una ventana de tamaño 100x100 pixeles.

Para la carga del Applet 3D en la página se separó en partes las capas del cuerpo humano y se adhirió otras ventanas y pestañas para su visualización, es decir, en una ventana cargará lo que es la capa de músculos y en otra cargará las capas piel, órganos, sistema respiratorio, huesos con el fin de no obtener el mismo error.

Programación en paralelo para los diferentes nodos del Clúster

El Applet de Java 3D pudo ser acoplado en una página HTML pero no cumple con el objetivo previsto de programación en paralelo para los diferentes nodos del Clúster, porque se trata de un Applet y este consume los recursos de la propia máquina a pesar de realizar la programación con hilos, por tal motivo se requiere seleccionar otra herramienta de programación junto a una IDE que ofrezca la flexibilidad para continuar con la investigación y avanzar con los procesos de paralelización del aplicativo 3D.

Luego de observar las ventajas que proporciona la biblioteca de OpenSceneGraph con programación distribuida y paralela se procede a realizar la elaboración de la programación en C++ con esta librería para un nuevo aplicativo 3D que permita cumplir con uno de los objetivos planteados en este documento.

No se realiza la programación con Java debido a que no se encuentra mucha información con esta biblioteca por el momento que pueda ser de ayuda para la elaboración del aplicativo 3D.

Prototipo 2. Diseño e implementación del aplicativo con OpenSceneGraph

Como en el prototipo anterior no se observó un avance significativo en el desarrollo del proyecto para ser paralelizado en los diferentes nodos del Clúster, se hizo necesario implementar un módulo para el manejo de los “.obj” construido en C++ junto a la librería OpenSceneGraph adecuada para la creación y manipulación de este tipo de entornos, que contará con algoritmos similares a los usados en Java 3D.

A. Grafo de escena

Es una estructura compuesta por nodos que contienen datos que definen un escenario virtual y controlan un proceso de dibujado. Tiene descripciones de bajo nivel de la geometría y apariencia visual de los objetos, así como descripciones de alto nivel referentes a la organización espacial de la escena, sirviendo como una especificación de documentación que permite representar gráficamente la información de los objetos del ambiente virtual.

El grafo de escena tiene como función

- Ayudar a crear una organización lógica de la escena.
- Establecer dependencias jerárquicas entre los distintos sistemas de referencia.
- Permitir el proceso automático de *culling* (eliminación automática de objetos).
- Facilitar el control de la escena por parte del usuario.

B. Tipos básico de nodos:

- **Node geometry:** Almacena la información poligonal de los objetos, también información referente a su apariencia como: material, textura, etc.
- **Node group:** Agrupa varios nodos hijos para facilitar el proceso de *culling* jerárquico.
- **Transform:** Clase base para aplicar una transformación al *subgrafo*.
- **MatrixTransform:** Tiene una matriz de 4x4 representando una transformación de operaciones con: *make translate*, *make rotate*, *make scale* y *preMult/postMult* para multiplicar matrices.

- **PositionAttitudeTransform:** Transformación que usa un Vector de tres coordenadas (Vec3) para la posición y una rotación de *Cuaternion* (Quat) para la actitud y un Vec3 para el escalado.
- **DOFTransform:** Nodo de transformación de grados de libertad.
- **Geode:** Es un nodo hoja que almacena la información geométrica de un objeto.
- **Billboard:** Rota una geometría de modo que siempre aparezca orientada hacia el punto de vista. Es especialmente útil para representar objetos que tienen una simetría axial, como pueden ser árboles, farolas, etc.
- **Node LOD (Nodo de nivel de detalle):** Selecciona uno de sus hijos, basándose en la distancia entre el objeto con múltiples niveles de detalle y el punto de vista. Se emplea para gestionar distintos niveles de detalle de un objeto.
- **Impostor:** Añade soporte para el cacheado jerárquico de imágenes.
- **Switch:** Es un nodo que puede tener varios hijos y permite que el usuario seleccione cuáles de ellos quiere que sean dibujados.
- **Sequence:** Es un nodo que puede tener varios hijos, los cuales van mostrando las imágenes de forma secuencial. Se utiliza para representar secuencias animadas. Una secuencia consiste en una lista ordenada de hijos, cada uno de los cuales tiene una duración asignada. Es posible hacer que la secuencia se ejecute de inicio a fin, de fin a inicio, que se repita cíclicamente, etc.
- **LightSource:** Posición un objeto *light* en la escena.
- **ClipNode:** Posiciona un objeto *clip plane* en la escena.
- **Projection:** Sobrecarga la matriz de proyección.
- **OccluderNode:** Permite colocar en la escena planos y cajas para definir oclusiones entre objetos.

La Figura 3-51 muestra la estructura para el sistema del aplicativo con OpenSceneGraph, el cual define el escenario gráfico que contiene un “*Graphics Context*” donde el “*Group*” es la raíz del sub-gráfico.

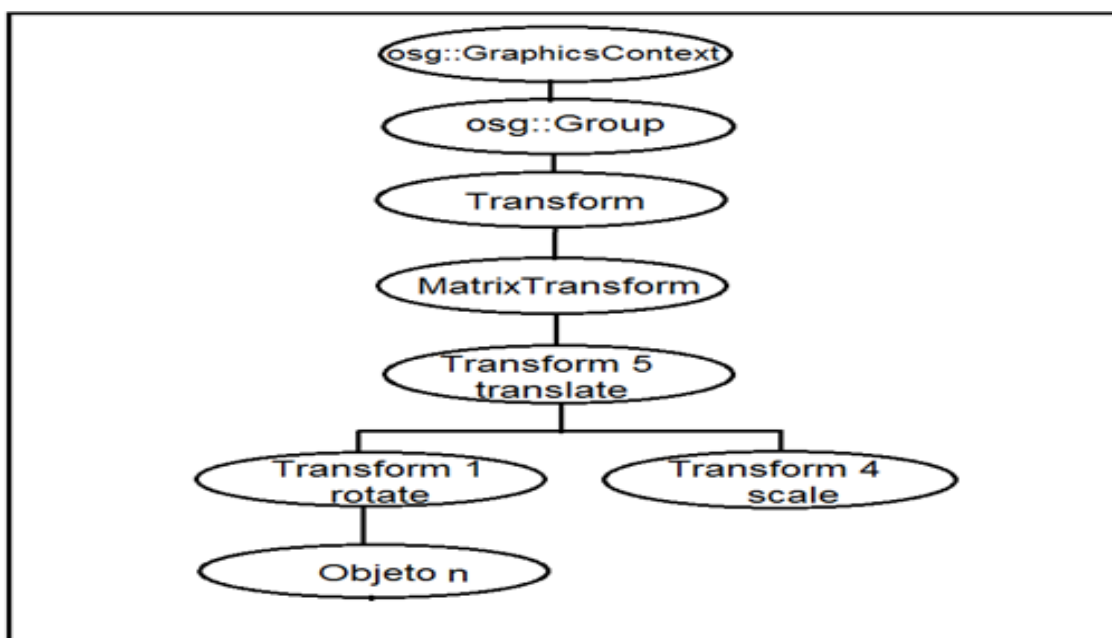


Figura 3-51 Estructura para el sistema del aplicativo con OpenSceneGraph

Fuente: Los Autores

Para la construcción del escenario gráfico se definió el grupo *Group* (*osg::Group*) que es el nodo que va a tener como hijos (*child*) a los objetos de un escenario gráfico para ser renderizados, el cual estará asociado al grupo *Transform* que aplica a todos los nodos hijos en el árbol por medio de *MatrixTransform* cualquier tipo de transformaciones (traslación, rotación, escala) que se requieran, donde se incorporará el *objeto n* que contendrá las formas 3D por medio de nodos (*osg::node*) que serán leídos desde una dirección con el nodo DB (*osg:DB*). El conjunto de nodos con los objetos se conectarán al PAT (PositionAttitudeTransform) para definir la posición en el espacio y junto al *Behavior* (*osg::GA*) especificarán la interacción con los objetos.

La librería de manipulación de objetos llamada “*osgManipulator*” está definida como un conjunto de “*dragger*”²⁸, “*selection*”²⁹ y “*commandManager*” y las órdenes que se generen serán por las interacciones del ratón con los “*dragers*”.

²⁸Dragger: Es un elemento gráfico que responde a las acciones que se realizan con el ratón sobre él para ejecutar ciertas órdenes específicas

²⁹Selection: Es el conjunto de objetos sobre los que actuarán las órdenes.

C. Casos de uso con notación UML.

A continuación se muestra el diagrama de uso general del sistema.

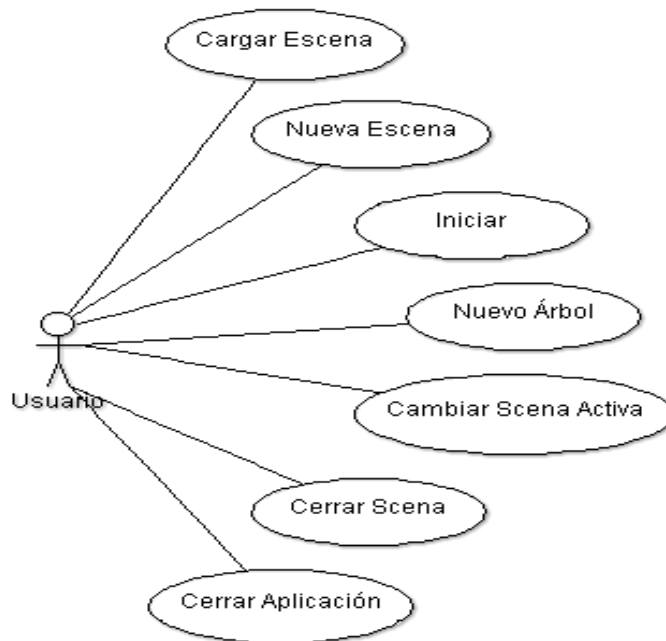


Figura 3-52 Diagrama casos de uso general del sistema

Fuente: Autores

Actor: Usuario

Actividad: Realiza el proceso de carga de escena, iniciar, cambio de escena, cierre de escena y cierre de la Aplicación.

Observaciones: Al ingresar a la escena se cargará de forma automática los objetos.

A continuación se define la acción que existe entre el usuario o cliente, se muestra el diagrama de uso general del sistema.

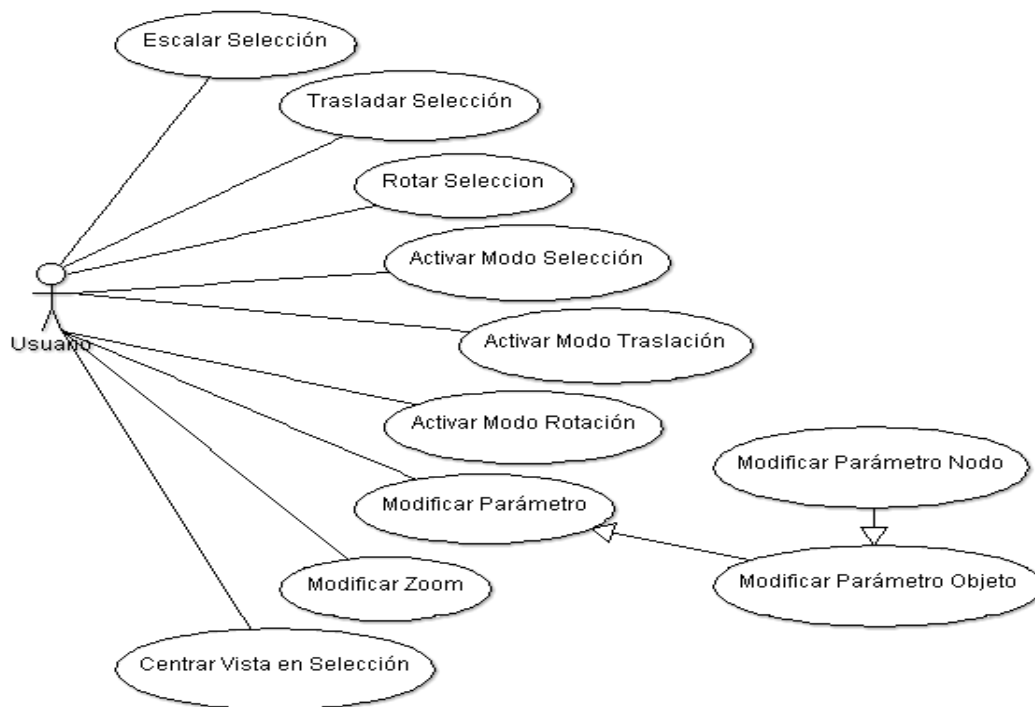


Figura 3-53 Diagrama casos de uso general del sistema

Fuente: Autores

Actor: Usuario

Actividad: Activa el proceso de trasladar, rotar, escalar para su transformación. También modifica los parámetros del nodo y del objeto.

Observaciones: Al ingresar a la escena los objetos 3D podrán ser manipulados con el mouse y el teclado.

Diseño

Iteración 1. Diseño e implementación del aplicativo con OpenSceneGraph

A. Diseño y diagrama de clases

El diseño de clases se basa en la orientación a objetos empleados en C++, se ha estructurado la aplicación en diferentes clases dependiendo de las distintas interacciones que se realizarán para el desarrollo del aplicativo.

Se estructura la solución en las siguientes clases:

Diagrama de clases para el aplicativo 3D con OSG		
Clase	Función	Diagrama
Principal	Se encuentra el método <i>main</i> que ejecuta la aplicación con MPI, contiene la estructura de las dimensiones de la pantalla y la llamada a los métodos para la visualización de la escena gráfica.	<pre> class Principal.cpp Matrix matrix; Double angle0; Double angle1; Double angle2; Matrix rot0; Matrix rot1; Matrix rot2; Vec3d ojo; Vec3d centrocara; Vec3d ejevert; Node* objeto; Group* root; PositionAttitudeTransform objetoXform; Vec3 objetoPosit; Integer val; virtual void Principal() virtual Group CreateScenemetodos() virtual int main(Integer argc, Char argv) </pre>
CreateModelToSaveVisitor	Clase de creación del objeto a ser guardado, ya sea por sección o completo.	<pre> class CreateModelToSaveVisitor : public Principal.cpp Scribe* scribe; Group* group; virtual void CreateModelToSaveVisitor() virtual void apply(Node node) </pre>
DeleteSelectedNodeVisitor	Clase de eliminación del objeto seleccionado con el pick.	<pre> class DeleteSelectedNodesVisitor : public Principal.cpp Node* node; Scribe* scribe; virtual void DeleteSelectedNodesVisitor() virtual void apply(Node node) virtual void pruneSelectedNodes() </pre>

Diagrama de clases para el aplicativo 3D con OSG		
Clase	Función	Diagrama
PickHandler	Clase de selección de objetos, se hace blanco el borde de todo el objeto para resaltar lo escogido.	<pre> class PickHandler : public Principal.cpp Double mx; Double my; Viewer* viewer; virtual void PickHandler() virtual bool handle(GUIEventAdapter ea, GUIActionAdapter aa) virtual void pick(GUIEventAdapter ea, void Viewer) virtual void toggleScribe(Group* parent, Node* node) virtual void saveSelectedModel(Node* scene) </pre>
osgsist_respiratorio	Contiene la estructura que llama a los objetos 3D para la capa del sistema respiratorio desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la librería Principal.h.	<pre> class ogsist_respiratorio.h : public Principal.cpp Integer a; Integer b; Integer c; String direccion; Node* objeto1; Node* objeto4; Group* root; PositionAttitudeTransform* objetoXform1; Vec3 objetoPosit1; PositionAttitudeTransform* objetoXform4; Vec3 objetoPosit4; virtual Node* CreateScenesist_respiratorio() virtual void ogsist_respiratorio() </pre>
osgpiel	Contiene la estructura que llama a los objetos 3D para la capa de piel desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la librería Principal.h.	<pre> class osgpiel.h : public Principal.cpp Integer a; Integer b; Integer c; String direccion; Node* objeto1; Node* objeto2; Group* root; PositionAttitudeTransform* objetoXform1; Vec3 objetoPosit1; PositionAttitudeTransform* objetoXform2; Vec3 objetoPosit2; virtual Node* CreateScenepiel() virtual void osgpiel() </pre>

Diagrama de clases para el aplicativo 3D con OSG		
Clase	Función	Diagrama
osgórganos	Contiene la estructura que llama a los objetos 3D para la capa de órganos desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la librería Principal.h.	<pre> class osgorganos.h : public Principal.cpp Integer a; Integer b; Integer c; String direccion; Node* objeto1; Node* objeto19; Group* root; PositionAttitudeTransform* objetoXform1; Vec3 objetoPosit1; PositionAttitudeTransform* objetoXform19; Vec3 objetoPosit19; virtual Node* CreateSceneorganos() virtual void osgorganos() </pre>
osgmúsculos	Contiene la estructura que llama a los objetos 3D para la capa de músculos desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la librería Principal.h.	<pre> class osgmusculos.h : public Principal.cpp Integer a; Integer b; Integer c; String direccion; Node* objeto1; Node* objeto129; Group* root; PositionAttitudeTransform* objetoXform1; Vec3 objetoPosit1; PositionAttitudeTransform* objetoXform129; Vec3 objetoPosit129; virtual Node* CreateScenemusculos() virtual void osgmusculos() </pre>
osgesqueleto	Contiene la estructura que llama a los objetos 3D para la capa de huesos desde la URL y los adhiere a la escena cuando sean llamados, está relacionada con la librería Principal.h.	<pre> class osgesqueleto.h : public Principal.cpp Integer a; Integer b; Integer c; String direccion; Node* objeto1; Node* objeto26; Group* root; PositionAttitudeTransform* objetoXform1; Vec3 objetoPosit1; PositionAttitudeTransform* objetoXform26; Vec3 objetoPosit26; virtual Node* CreateScenehuesos() virtual void osgesqueleto() </pre>

Tabla 3-23 Lista de clases OSG

Fuente: Autores

El Diagrama de clases para el aplicativo 3D realizado con la librería OSG tiene la clase principal, la cual va a contener al *main* que llamará a la librería *principal.h* donde se encuentra integrado las llamadas a las clases para la manipulación con el teclado, el mouse, para la selección y eliminación del objeto y la llamada a las librerías que contendrán los órganos del cuerpo humano en 3D (ver anexo 5.5.2.4).

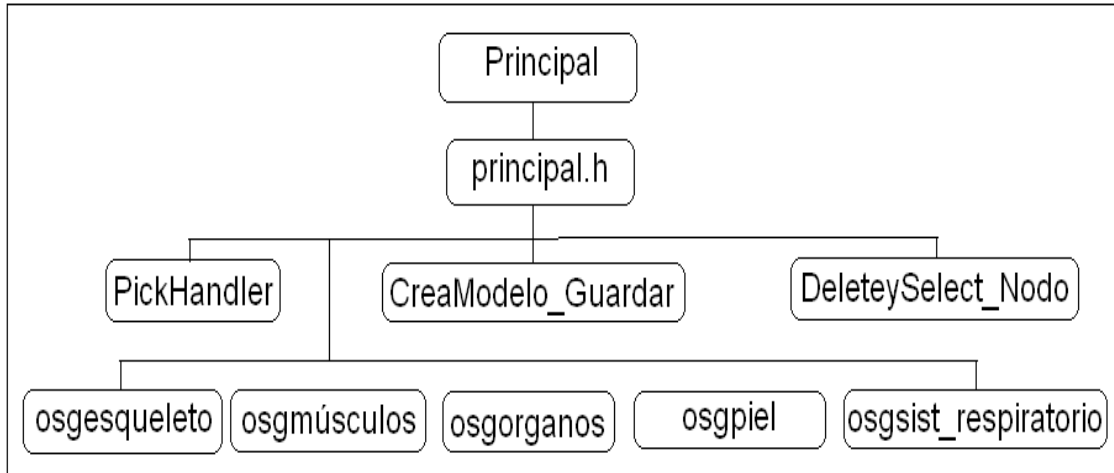


Figura 3-54 Diseño de Clases OSG.

Fuente: Autores

B. Descripción de librerías

Para iniciar con la programación 3D se debe realizar enlaces a las librerías de OSG que servirán para elaborar el aplicativo. Con estas se podrá visualizar un objeto, su color y coordenadas en las que aparecerá. También efectuarán la lectura de ficheros de escena 3D que servirá para cargar los archivos .obj, así como la utilización de teclado:

Librerías OSG para visualización
<pre> #include<osgViewer/Viewer>//visualiza las imágenes #include<osg/Node>//librería para cargar nodos #include<osgDB/ReadFile>//librería para leer archivos #include<osg/PositionAttitudeTransform>//librería para colocar en una posición #include<osg/io_utils> #include<osg/MatrixTransform>//librería para utilizar matrices #include<osgFX/Scribe> #include<osgGA/TrackballManipulator> #include<osgGA/StateSetManipulator> #include <osgDB/WriteFile>//librería para escribir archivos #include <osg/Notify>//librería para generar notificaciones </pre>

Tabla 3-24 Librerías OSG

Fuente: Autores

La lectura y manipulación de objetos han sido implementadas en librerías que serán referenciadas como se muestra a continuación:

Librerías OSG para manipulación
<pre> #include "Principal.h" //librería para la lectura de todas las librerías de las capas del cuerpo humano e iteraciones con los dispositivos de entrada del usuario. #include "osgesqueleto.h" // librería para la lectura de la capa esqueleto #include "osgpiel.h"// librería para la lectura de la capa piel #include "osgorganos.h"// librería para la lectura de la capa órganos #include "osgsist_respiratorio.h"// librería para la lectura de la capa sist. Respiratorio #include "osamusculos.h"//librería para la lectura de la capa músculos </pre>

Tabla 3-25 Librerías OSG

Fuente: Autores

Codificación

Iteración 1. Diseño e implementación del aplicativo con OpenSceneGraph

Capa: Piel

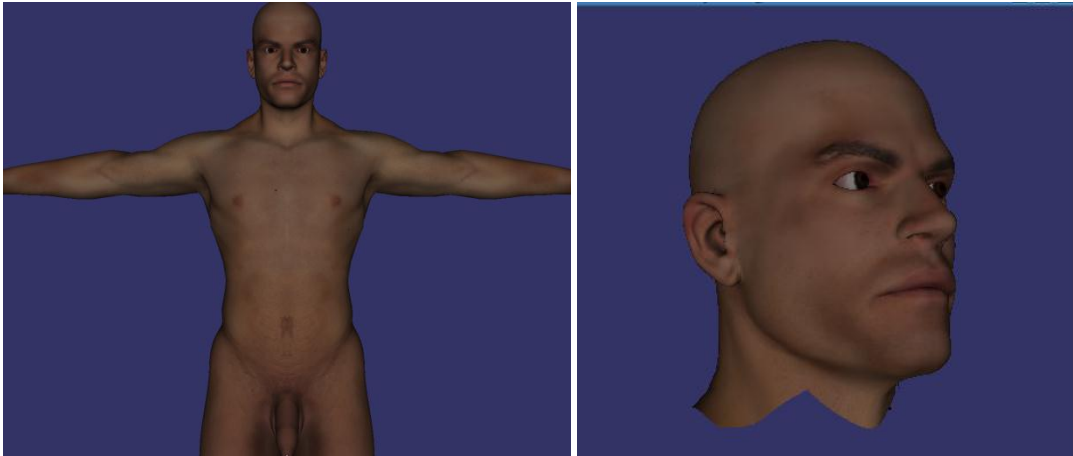


Figura 3-55 Ventana de la capa de piel

Fuente: Autores

Módulo: Librería propia de capa piel

Pantalla: Aplicativo capa piel

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa de piel será la primera en ser visualizada y retirada al seleccionar y dar clic.

Parte del código se describe a continuación:

Como primera instancia se utiliza el espacio de nombre *using*

```
using namespace std;  
using namespace osg;  
using namespace osgViewer;
```

Método para llamar a los objetos para la capa piel desde la URL, se crea la variable con la dirección en la que se encuentra el archivo que contiene a las imágenes y los nodos para cargar las imágenes, también se crea la posición del objeto en la escena

```
Node* CreateScenepiel(){
Objeto1 = osgDB::readNodeFile( direccion + "/Model_Skin/cuerpox.obj" );
Group* root = newGroup();
PositionAttitudeTransform* objetoXform = newPositionAttitudeTransform();
Vec3objetoPosit(a,b,c);
objetoXform ->setPosition(objetoPosit);
```

Posición de los nodos en el root

```
root->addChild(objetoXform); ///añade el nodoal root conlaposicióndadaen el
PositionAttitudeTransform.
/// Añade las imágenes en sus respectivas posiciones antes dadas
objetoXform ->addChild(objeto1);
return root;
}
```

Capa: Músculos

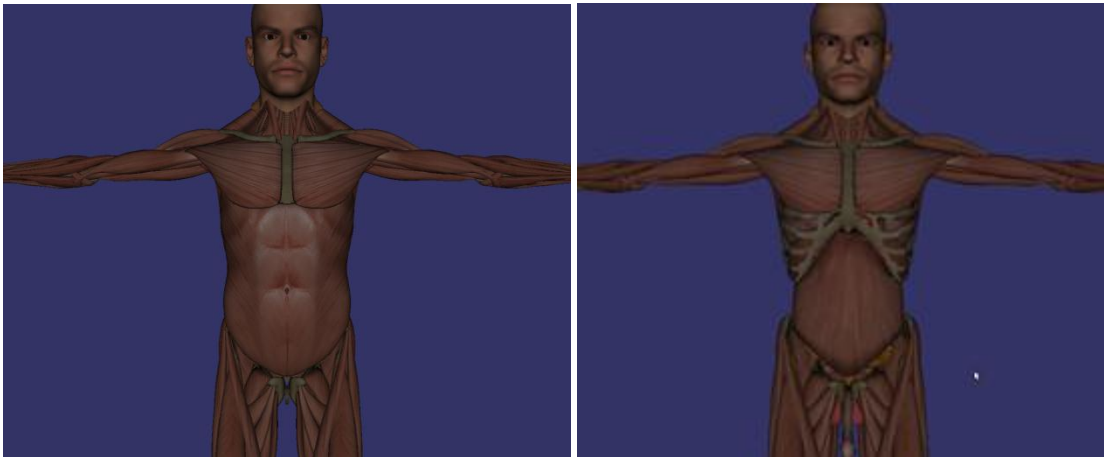


Figura 3-56 Ventana de la capa músculos

Fuente: Autores

Módulo: Librería propia de capa músculos

Pantalla: Aplicativo capa músculos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: posicionar puntero del mouse y clic

Descripción: La capa de músculos será la siguiente en ser visualizada después de retirar la capa piel y cada objeto elegido se retirará al seleccionar y dar clic.

Parte del código se describe a continuación:

Se utiliza el espacio de nombre *using*

Se establece el método para llamar a los objetos para la capa músculos desde la URL

Se crea la variable con la dirección en la que se encuentra el archivo que contiene a las imágenes y los Nodos para cargar las imágenes, también se crea la posición del objeto en la escena

```
class osgmusculos {
public:
    osg::Node* CreateScenemusculos() {
        Model_Muscular_Abductor_Digiti_Minimi = osgDB::readNodeFile( direccion
        + "/Model_Muscular/Model_Muscular_Abductor_Digiti_Minimi.obj" );
        Model_Muscular_Abductor_Hallucis = osgDB::readNodeFile( direccion +
        "/Model_Muscular/Model_Muscular_Abductor_Hallucis.obj" );
        Group* root = new Group();
        PositionAttitudeTransform* objetoXform = new PositionAttitudeTransform();
        Vec3objetoPosit(a,b,c);
        objetoXform ->setPosition(objetoPosit);
    }
};
```

Se indica la posición de los nodos y se añade al *root* con la posición dada en el *PositionAttitudeTransform*.

```
root->addChild(objetoXform);
root->addChild(objetoXform2);
```

Añade las imágenes en sus respectivas posiciones antes dadas

```
objetoXform ->addChild(Model_Muscular_Abductor_Digiti_Minimi);
objetoXform2 ->addChild(Model_Muscular_Abductor_Hallucis);
return root;}}
```

Capa: Órganos

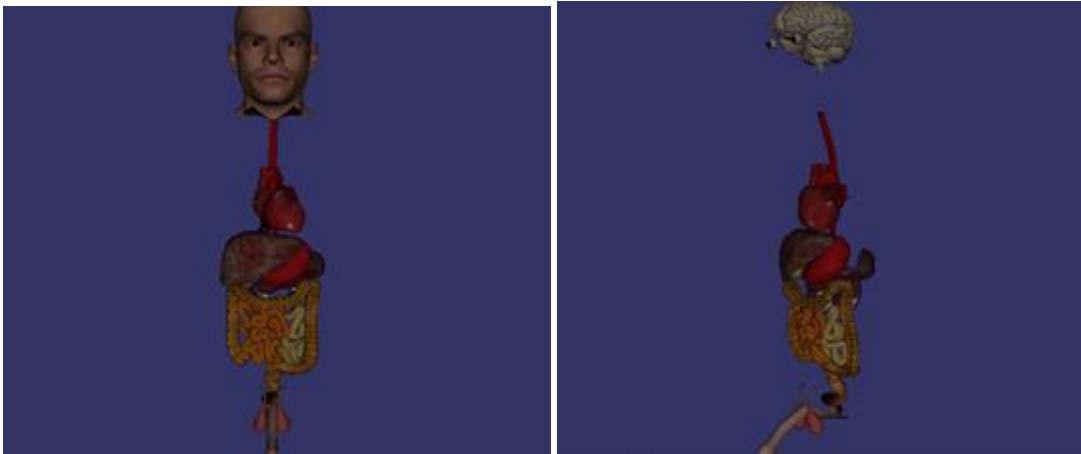


Figura 3-57 Ventana de la capa órganos

Fuente: Autores

Módulo: Capa órganos

Pantalla: Aplicativo capa órganos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa de órganos será la siguiente en ser visualizada después de retirar algunos objetos de la capa músculos y cada elemento elegido se retirará al seleccionar y dar clic.

Parte del código se describe a continuación:

Se utiliza el espacio de nombre *using*

En el método para llamar a los objetos para la capa órganos desde la URL se crea la variable con la dirección en la que se encuentra el archivo que contiene a las imágenes y los Nodos para cargar las imágenes, también se crea la posición del objeto en la escena

```
class osgorganos {
public:
    osg::Node* CreateSceneorganos() {
        baso = osgDB::readNodeFile( direccion + "/Model_Organ/baso.obj" );
        bilis = osgDB::readNodeFile( direccion + "/Model_Organ/bilis.obj" );
        Group* root = new Group();
        PositionAttitudeTransform* objetoXform = new PositionAttitudeTransform();
        Vec3objetoPosit(a,b,c);
        objetoXform ->setPosition(objetoPosit);
        PositionAttitudeTransform* objetoXform2 = new PositionAttitudeTransform();
        Vec3objetoPosit2(a,b,c);
        objetoXform2 ->setPosition(objetoPosit2);
```

Se indica la posición de los nodos y se añade al root con la posición dada en el *PositionAttitudeTransform*.

```
root->addChild(objetoXform);
root->addChild(objetoXform2);
```

Añade las imágenes en sus respectivas posiciones antes dadas

```
objetoXform ->addChild(baso);
objetoXform2 ->addChild(bilis);
return root;}}
```

Capa: Sistema Respiratorio

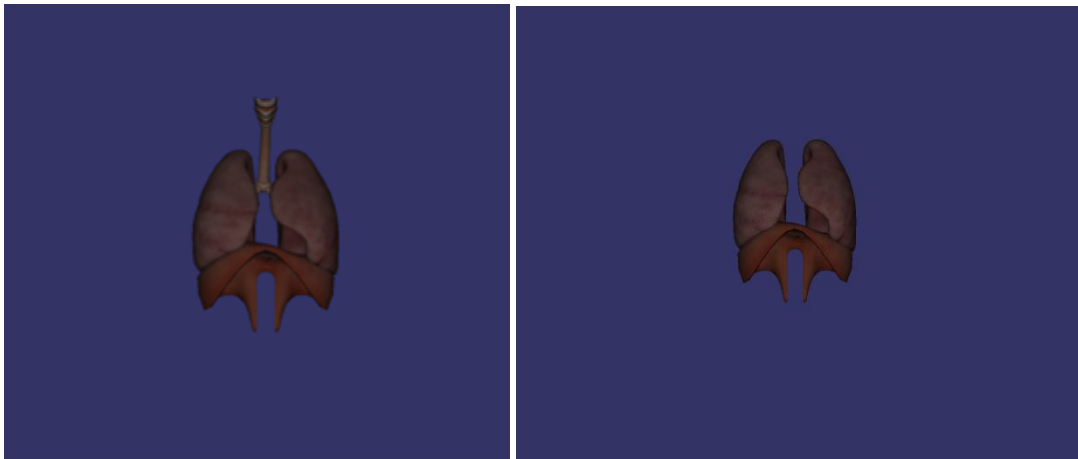


Figura 3-58 Ventana de la capa sistema respiratorio

Fuente: Autores

Módulo: Capa Sistema Respiratorio

Pantalla: Aplicativo capa Sistema Respiratorio

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa de sistema respiratorio será la siguiente en ser visualizada después de retirar algunos objetos de la capa músculos y cada elemento elegido se retirará al seleccionar y dar clic.

Se utiliza el espacio de nombre *using*

En el método para llamar a los objetos para la capa del sistema respiratorio desde la URL se crea la variable con la dirección en la que se encuentra el archivo que

contiene a las imágenes y los Nodos para cargar las imágenes, también se crea la posición del objeto en la escena

```
class osgsist_respiratorio {
public:
osg::Node* CreateScenesist_respiratorio(){
objeto1 = osgDB::readNodeFile( direccion
+ "/Model_Respiratory/Model_Respiratory_Epiglotis.obj" );
objeto2 = osgDB::readNodeFile(direccion +
"/Model_Respiratory/Model_Respiratory_Diaphragm.obj" );
Group* root = new Group();
PositionAttitudeTransform* objetoXform = new PositionAttitudeTransform();
Vec3objetoPosit(a,b,c);
objetoXform ->setPosition(objetoPosit);
PositionAttitudeTransform* objetoXform2 = new PositionAttitudeTransform();
Vec3objetoPosit2(a,b,c);
objetoXform2 ->setPosition(objetoPosit2);
```

Se indica la posición de los nodos y se añade al *root* con la posición dada en el *PositionAttitudeTransform*.

```
root->addChild(objetoXform);
root->addChild(objetoXform2);
```

Añade las imágenes en sus respectivas posiciones antes dadas

```
objetoXform ->addChild(objeto1);
objetoXform2 ->addChild(objeto2);
return root;}}
```


Capa: Esqueleto



Figura 3-59 Ventana de la capa esqueleto

Fuente: Autores

Módulo: Capa esqueleto

Pantalla: Aplicativo capa esqueleto

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar puntero del mouse y clic

Descripción: La capa esqueleto será la siguiente en ser visualizada después de retirar algunos objetos de las capas de: músculos y órganos, cada elemento elegido se retirará al seleccionar y dar clic.

Parte del código se describe a continuación:

Se utiliza el espacio de nombre *using*

En el método para llamar a los objetos para la capa huesos desde la URL se crear la variable con la dirección en la que se encuentra el archivo que contiene a las imágenes y los Nodos para cargar las imágenes, también se crea la posición del objeto en la escena

```

class osgesqueleto {
public:
    osg::Node* CreateScenehuesos() {
        craneoo1 = osgDB::readNodeFile(direccion + "/Model_Skeletal/craneoo1.obj" );
        fibula = osgDB::readNodeFile( direccion + "/Model_Skeletal/fibula.obj" );
        Group* root = newGroup();
        PositionAttitudeTransform* objetoXform = newPositionAttitudeTransform();
        Vec3objetoPosit(a,b,c);
        objetoXform ->setPosition(objetoPosit);
        PositionAttitudeTransform* objetoXform2 = newPositionAttitudeTransform();
        Vec3objetoPosit2(a,b,c);
        objetoXform2 ->setPosition(objetoPosit2);
    }
}

```

Se indica la posición de los nodos y se añade al *root* con la posición dada en el *PositionAttitudeTransform*.

```

root->addChild(objetoXform);
root->addChild(objetoXform2);

```

Añade las imágenes en sus respectivas posiciones antes dadas

```

objetoXform ->addChild(objeto1);
objetoXform2 ->addChild(objeto2);
return root;});

```

Capa: Principal

Módulo: Sistema Principal

Proceso: Ejecuta el inicio del sistema principal para la creación de la escena.

Descripción: La clase principal será la que inicialice el sistema mediante la creación del universo virtual y la llamada a escena al “*group*”.

Mediante la clase base “*osgGA::GUIEventHandler*” que está diseñada para definir acciones personalizadas para el teclado y la interfaz gráfica de usuario con los eventos del ratón se crearán acciones personalizadas para la manipulación de los objetos en la escena.

Estas librerías permiten visualizar al objeto y llamar a los métodos que tienen las capas del cuerpo humano.

```
#include<osgViewer/Viewer>
#include"Principal.h"
```

En la clase *main* se crea una ventana para visualizar a los objetos en la escena y no se muestren en toda la pantalla asignando el tamaño de la misma.

```
int main(int argc, char **argv ) {
    osg::ref_ptr<osg::GraphicsContext::Traits> traits =
    new osg::GraphicsContext::Traits;
    traits->x = 200;
    traits->y = 200;
    traits->width = 800;
    traits->height = 600;
    traits->windowDecoration = true;
    traits->doubleBuffer = true;
    traits->sharedContext = 0;
```

Se crea el *osgViewer* para la visualización del objeto en la escena

```

osg::ref_ptr<osgViewer::Viewer> viewer = new osgViewer::Viewer;
osg::ref_ptr<osg::GraphicsContext> gc =
osg::GraphicsContext::createGraphicsContext(trails.get());

gestionnaireEvenements* evenements = new gestionnaireEvenements();
osg::ref_ptr<KeyboardModel> keyboardModel = new KeyboardModel;
viewer->getCamera()->setGraphicsContext(gc.get());
viewer->getCamera()->setViewport(0,0,800,600);

```

Configura y entra en un bucle de simulación.

```

viewer->setSceneData(CreateScenemetodos());
osg::ref_ptr<osgGA::StateSetManipulator> statesetManipulator =
new osgGA::StateSetManipulator(viewer->getCamera()->getStateSet());

```

Agrega el controlador de eventos a la lista

```

viewer->addEventHandler(statesetManipulator.get());
viewer->addEventHandler(new PickHandler());
viewer->setCameraManipulator(new osgGA::TrackballManipulator());
viewer->realize();

```

Manda a llamar y visualiza al objeto varias veces

```

while(!viewer->done()) {
viewer->run();//funciona solo para telado
viewer->frame(); }
return 0; }

```

Clase principal.h

Permite unir y enlazar todas las clases y librerías al *main* o capa principal. Contiene el método que llama a las distintas capas del cuerpo humano.

Parte del código se describe a continuación:

CreateScenemetodos() es el método que crea las instancias para llamar a las librerías que contienen los objetos donde se encuentran las partes del cuerpo humano, además guarda todo lo que se encuentre en el grupo (*root*) en el lugar especificado.

```
osg::Group* CreateScenemetodos() {
unionmetodos1=piel.CreateScenepiel();//piel
transform1->addChild(unionmetodos1);//objeto a ser rotado
transform5->addChild(transform1); //objeto a ser zoom y a trasladar
root->addChild(transform5);
bool
result2=osgDB::writeNodeFile(*root,"/home/user3d/Final_codigo_OSG/osgbody/body1.osg"); //guarda
return root;}
```

La clase *Pick* y *Handle* permiten la manipulación de los objetos y recogen los eventos dentro de la escena, la siguiente tabla muestra el detalle que hará cada tecla al ser presionada.

Eventos del teclado	
Evento	Detalle
KEY_H	Guardará solo el objeto seleccionado con extensión .osg
KEY_O	Vuelve el grupo de objetos al origen en la posición 0
KEY_G	Guarda todo el grupo de objetos con extensión .osg dentro de la carpeta del proyecto.
KEY_Delete	Realiza la eliminación del objeto seleccionado previamente
KEY_Left	Traslada en forma horizontal en el eje negativo

Eventos del teclado	
Evento	Detalle
KEY_Right	Traslada en forma horizontal en el eje positivo
KEY_Down	Traslada en forma vertical en el eje negativo
KEY_Up	Traslada en forma vertical en el eje positivo
KEY_D	Rota en el eje X positivo
KEY_A	Rota en el eje X negativo
KEY_E	Realiza el zoom en el eje Y aumentando
KEY_Q"	Realiza la disminución del zoom en el eje Y

Tabla 3-26 Eventos del teclado para manipulación de los objetos 3D

Fuente: Autores

```

class PickHandler : public osgGA::GUIEventHandler{
public:
    PickHandler():
    ~PickHandler() {}
    bool handle(const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa)
    {
        osgViewer::Viewer* viewer = dynamic_cast<osgViewer::Viewer*>(&aa);
        if (!viewer) return false;
        switch(ea.getEventType()){
        case (osgGA::GUIEventAdapter::KEYUP):{
            if(ea.getKey()== 'h') {...}
            elseif(ea.getKey()== 'o') {
                ...
                Node * objeto1 =
                osgDB::readNodeFile("/home/user3d/Final_codigo_OSG/osgbody/body1.osg");
            }
            elseif(ea.getKey()== 'g') {
                ...
                osgDB::writeNodeFile(*(viewer->getSceneData()), "bodyall.osg");
            }
            elseif(ea.getKey()== osgGA::GUIEventAdapter::KEY_Delete) {...}
            return false;
        }
    }
}

```

```
case(osgGA::GUIEventAdapter::KEYDOWN){  
    switch(ea.getKey()) {  
        case(osgGA::GUIEventAdapter::KEY_Left): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_Right): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_Down): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_Up): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_D): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_A): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_E): ...  
        returnfalse;  
        break;  
        case(osgGA::GUIEventAdapter::KEY_Q): ...  
        returnfalse;  
        break;  
        default:  
    returntrue;  
    }  
}
```

Capa: Selección de Objetos



Figura 3-60 Ventana de la capa de selección de objetos

Fuente: Autores

Módulo: Capa de selección de objetos

Pantalla: Aplicativo capa de selección de objetos

Proceso: Ejecuta el código al dar clic en el objeto con el mouse en el escenario gráfico

Evento: Clic con el puntero del mouse

Descripción: La capa de selección de objetos actuará en el momento en que el cursor del mouse se posicione sobre cualquier objeto y de clic, con lo que inmediatamente se tornará de un color blanco estando listo para realizar cualquier acción posterior al dar clic.

Utilizando la librería *osgGA/GUIEventHandler* se creará un comportamiento que espere eventos del ratón en el escenario gráfico, este permitirá ejecutar el código específico del objeto seleccionado.

Parte del código se describe a continuación:

Se crea variables para saber el lugar de selección del objeto


```
protected:
float _mx,_my;
bool _usePolytopeIntersector;
bool _useWindowCoordinates;
```

En el método *toggleScribe* si no existe un nodo seleccionado retorna el Padre y agrega si es otro objeto el escogido.

```
void toggleScribe(osg::Group* parent, osg::Node* node , osgViewer::Viewer*
viewer) {
if (!parent || !node) return;
    osgFX::Scribe* parentAsScribe = dynamic_cast<osgFX::Scribe*>(parent);
if (!parentAsScribe) {
    // node aun no pinchado, entonces destaca con un osgFX::Scribe
    osgFX::Scribe* scribe = new osgFX::Scribe();
    scribe->addChild(node);
    parent->replaceChild(node,scribe);
} } }
```

Método para guardar un objeto seleccionado

```
void saveSelectedModel(osg::Node* scene) {
if (!scene) return; //Si no hay objeto seleccionado no guarda nada.
CreateModelToSaveVisitor cmtsv;
scene->accept(cmtsv);
if (cmtsv._group->getNumChildren()>0) {
    std::cout<<"Archivo guardado 'objeto3d.osg'"<<std::endl;
    osgDB::writeNodeFile(*cmtsv._group, "objeto3d.osg");
} };
```

Capa: Eliminación de Objetos

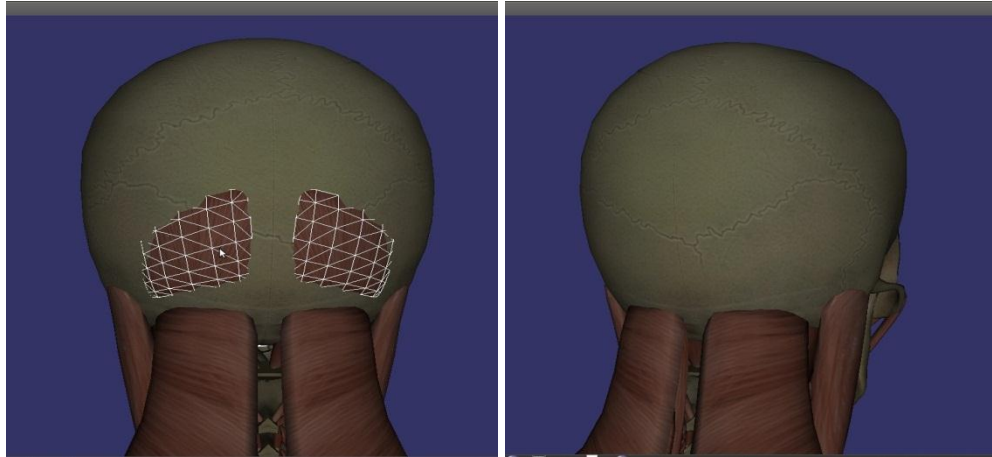


Figura 3-61 Ventana de la capa de eliminar objetos

Fuente: Autores

Módulo: Capa de eliminar objetos

Pantalla: Applet capa de eliminar objetos

Proceso: Ejecuta el código al dar clic en el escenario gráfico

Evento: Posicionar y clic con el puntero del mouse

Descripción: La capa de eliminar objetos actuará en el momento en que el puntero del mouse sea presionado sobre cualquier objeto luego de haber sido posicionado, seleccionado y puesto de color blanco, con lo que inmediatamente será quitado de la escena, quedando listo para la siguiente acción.

Utilizando la librería *osgGA/GUIEventHandler* se creará un comportamiento que espere eventos del ratón en el escenario gráfico, este permitirá ejecutar el código específico del objeto seleccionado para su eliminación.

Parte del código se describe a continuación:

La clase *DeleteSelectedNodesVisitor* realiza la eliminación de objetos cuando es llamado.

```

Class DeleteSelectedNodesVisitor :public osg::NodeVisitor{
public: DeleteSelectedNodesVisitor():
    osg::NodeVisitor(osg::NodeVisitor::TRAVERSE_ALL_CHILDREN) {}
virtual void apply(osg::Node& node) {
    osgFX::Scribe* scribe = dynamic_cast<osgFX::Scribe*>(&node);
    if (scribe) { _selectedNodes.push_back(scribe); }
    else { traverse(node); } }

```

Si efectúa el evento clic llama a la clase *DeleteSelectedNodesVisitor* con lo cual se elimina el objeto previamente seleccionado

```

osg::notify(osg::NOTICE)<<"Delete"<<std::endl;
DeleteSelectedNodesVisitor dsnv;
viewer->getSceneData()->accept(dsnv);
dsnv.pruneSelectedNodes();

```

Pruebas con OSG

Las imágenes 3D editadas se añaden al aplicativo con OSG para posteriormente ser seleccionadas y eliminadas iniciando desde la capa piel luego la capa músculos, los órganos, sistema respiratorio y huesos dando como resultado la siguiente imagen:

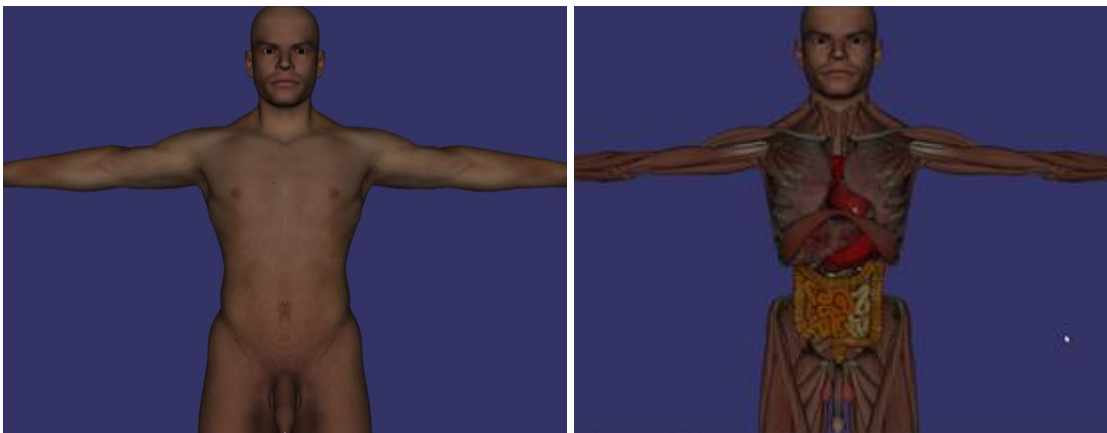


Figura 3-62 Ventana de aplicativo con cuerpo humano capas piel, músculos, órganos, sistema respiratorio y huesos

Fuente: Autores

Comprobación de errores

Error 01: El objeto 3D no se visualiza en el escenario gráfico

En el Archivo “MTL” (*.mtl*) (ver Figura 4.8) que contiene las declaraciones que describen los materiales, se encuentra la declaración *Tr* creada al momento de guardar desde 3D Max Studio la cual no permite que se visualice el objeto en el escenario gráfico.

Error 02: Las imágenes de las texturas de los objetos no se visualizan

En el Archivo “MTL” (*.mtl*) (ver Figura 3-63) se encuentra la declaración *map_Kd* que contiene el nombre del archivo de imagen con su dirección, al ejecutar el aplicativo con OSG este no es leído y no coloca la textura dando como resultado la imagen en gris.

```
# 3ds Max Wavefront OBJ Exporter v0.97b - (c)2007guruware
newmtl01___Default
Ns 9.999999046326
Ni 1.500000000000
d 1.000000000000
    Tr 0.000000000000
    Tf 1.000000000000 1.000000000000 1.000000000000
    illum 2
    Ka 0.588235318661 0.588235318661 0.588235318661
    Kd 0.588235318661 0.588235318661 0.588235318661
    Ks 0.000000000000 0.000000000000 0.000000000000
    Ke 0.000000000000 0.000000000000 0.000000000000
    map_Kd
C:\Users\maopc\Desktop\imagesphsh\Model_Muscular_Zygomaticus_Minor.jpg
```

Figura 3-63 Archivo “MTL” (*.mtl*) guardado desde 3D Max Studio del archivo de imagen Model_Muscular_Zygomaticus_Minor

Fuente: Autores

Solución de errores

Solución 01: El objeto 3D no se visualiza en el escenario gráfico

La declaración *Tr* deberá ser eliminada de todos los archivos *.mtl* para que el objeto pueda ser visualizado en el escenario gráfico.

Solución 02: Las imágenes de las texturas de los objetos no se visualizan

La carpeta en la cual se ejecuta el aplicativo deberá contener los archivos de las imágenes (*.obj*, *.mtl*, *.jpg*) y en la declaración *map_Kd* se deberá borrar la dirección dejando el nombre del archivo de imagen antepuesto por el “*slash*” por ejemplo *map_Kd /Model_Muscular_Zygomaticus_Minor.jpg*, con esto será posible visualizar la textura en el escenario gráfico.

Iteración 2. Implementación de Procesos Paralelos y Distribuidos en una Aplicación 3D mediante OSG

Para realizar la comunicación con el clúster y el aplicativo OSG se debe recurrir a otras de las librerías que proporciona OSG con lo cual se realizará la programación necesaria, adjuntando además código MPI el cual no es complejo, pero es relativamente distinta a la programación de un código serial y existen factores nuevos a tener en cuenta.

Se debe incluir el encabezado *<mpi.h>*

Se debe inicializar y terminar MPI con las funciones *MPI_Init* *MPI_Finalize* respectivamente.

El código entre estas llamadas será ejecutado simultáneamente por todos los procesadores.

Fuera de ese lapso no está definido el comportamiento del programa (depende de la implementación de MP).

La forma de comunicación en MPI es a través de mensajes que contienen datos.

La forma más simple es la comunicación punto a punto, donde se envía un mensaje de un proceso a otro.

Esto se realiza usando las funciones *MPI Send* y *MPI Recv*.

Diseño

Aplicativos desarrollados incluidos en el Clúster con MPI

El diseño de MPI está inspirado en máquinas con una arquitectura de memoria distribuida donde cada procesador es propietario de cierta memoria y la única manera de intercambiar información es a través de mensajes.

Sin embargo también se encuentran implementaciones de MPI en máquinas de memoria compartida

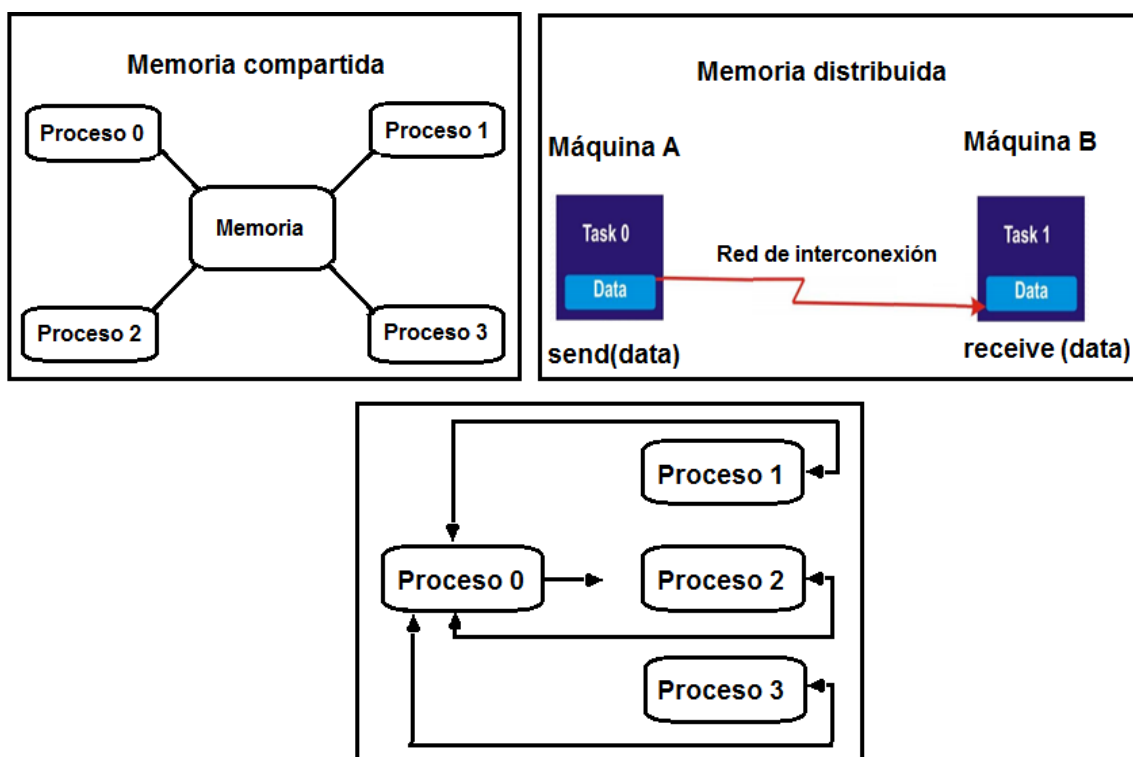


Figura 3-64 Diseño de un modelo paralelo

Fuente: Fialho Leonardo, Incorporando RADICa OpenMPI, 2012

Codificación

Librerías a utilizar para la programación en el Clúster y código MPI para la paralelización del aplicativo en los nodos del clúster.

```
#include<osg/Group>
#include<osgDB/Registry>
#include<osgDB/ReadFile>
#include<osgGA/TrackballManipulator>
#include<osgGA/StateSetManipulator>
#include<osgViewer/Viewer>
#include<osgViewer/ViewerEventHandlers>
#include<osg/Quat>
#include<osg/io_utils>
```

```
#include"receiver.h"
#include"broadcaster.h"
#include"Principal.h"
#include<mpi.h>
```

El código principal de OpenMPI es implementado en la clase *main* de la programación con osgclúster donde se indicará que debe ejecutar el maestro (FrontEnd) y que los esclavos (nodos) para inicializar el ambiente de ejecución paralelo.

Para la visualización de los objetos en el maestro y en los esclavos se requiere el uso y manejo de la cámara y la conversión de los datos para la lectura y escritura en los *frames*.

En el *main* se realizará la llamada al método que contiene los objetos para su incorporación en la escena además de colocar los códigos MPI para la visualización en los nodos:

```
int main( int argc, char **argv ){
int size,rank; // variables de tamaño y rango usados en el paso de mensajes
MPI_Init(&argc, &argv); // inicializa el proceso de paso de mensajes
```

Determinar el identificador (*rank*) del proceso actual

```
MPI_Comm_rank( MPI_COMM_WORLD, &rank ); //Identificación dentro del
ambiente paralelo
```

Obtener el número de procesos que realmente se pudo iniciar

```
MPI_Comm_size( MPI_COMM_WORLD, &size ); // devuelve el número de
procesos en este COMM_WORLD
```

Cuando se manda a cargar la escena se realiza en el maestro una manipulación especial para trabajar con el cluster tomando a la cámara en la posición cero como guía además se realiza una sincronización y en los esclavos se le dice que si hay una interrupción en la lectura de los datos se mande a apagar al maestro

```
while( !viewer->done() && !masterKilled ) {
viewer->advance();///  
switch (viewerMode) {  
case(MASTER): {  
osg::Matrix modelview(viewer->getCamera()->getViewMatrix());///  
...  
bc.sync();}  
break;  
case(SLAVE): {  
rc.sync();  
...  
}}
```



```

if (cp->getMasterKilled()){ masterKilled = true; } }
break;
default:
break; }

```

Actualiza y llama a los nodos a un redibujado de la escena

```

...
viewer->eventTraversal();///
viewer->updateTraversal();///
if (viewerMode==SLAVE) { ... }
if(!masterKilled)viewer->renderingTraversals(); }
...
if (viewerMode==MASTER) { ... }

```

Finalizar el ambiente de ejecución paralelo

```

...
Printf("Soy el procesador %d de un total de %d\n",size,rank);
MPI_Finalize(); // Finaliza el proceso de paso de mensajes
return 0;}

```

El proceso de compilación se lo puede revisar en Anexo Proceso de compilación con MPI.

Pruebas con MPI

Se utiliza un Clúster de visualización y de igual manera nodos de visualización para la ejecución del aplicativo 3D.

El aplicativo osgclúster con programación MPI se visualizó en el FrontEnd y en los nodos (tile) como muestra la Figura 3-65 .

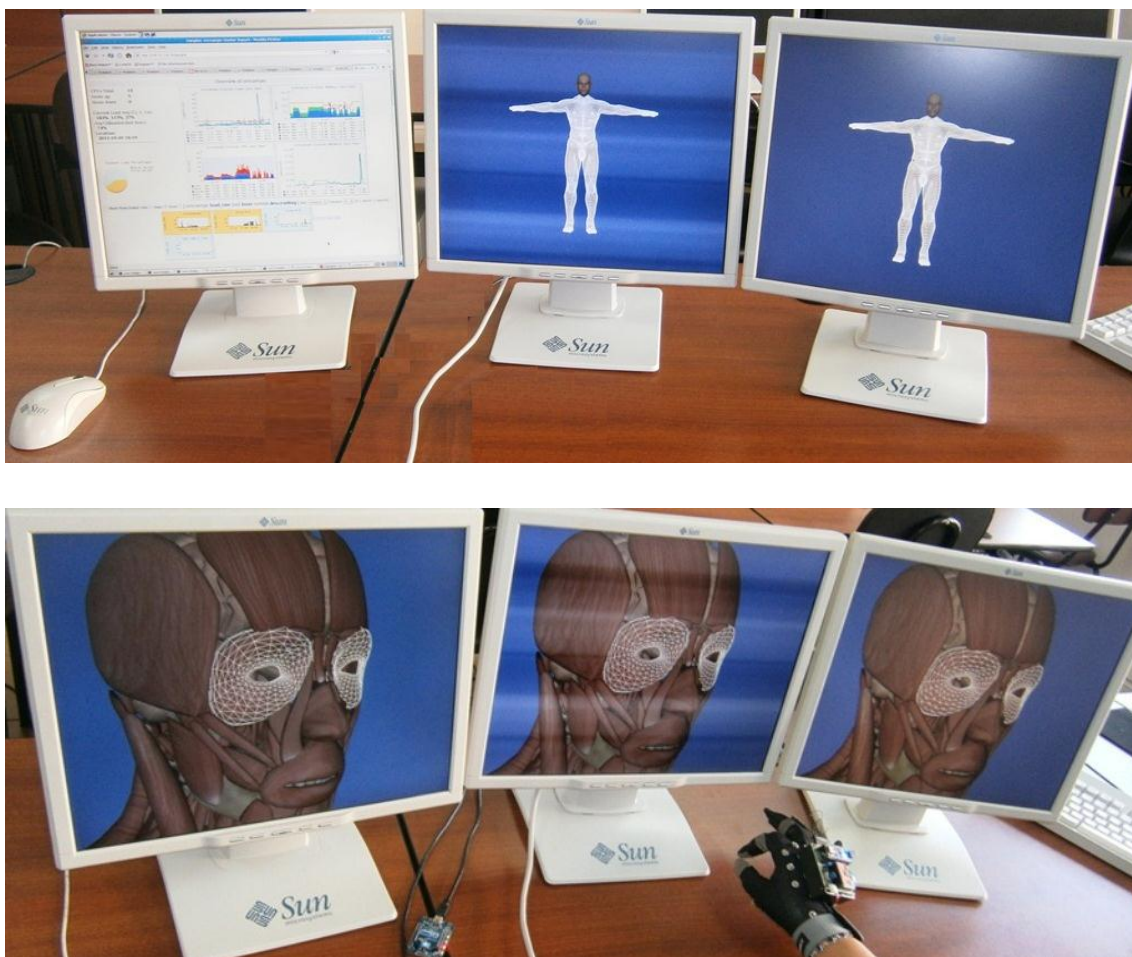


Figura 3-65 Ventana de visualización con MPI en el FrontEnd y en los nodos

Fuente: Autores

Comprobación de errores

Error 01: La visualización se realiza en el FrontEnd o donde se ejecute la aplicación.

Al ejecutar el aplicativo en una máquina como nodo maestro esta no la distribuye ni la paraleliza en los nodos esclavos.

Error 02: La herramienta Ganglia no muestra el procesamiento que se está realizando

En Ganglia se visualiza la carga de trabajo en el FrontEnd y en un solo nodo al momento de la ejecución, pero no muestra saturación de procesamiento en los diagramas de registro de eventos solo un bajo proceso.

Solución de errores

Solución 01: La visualización se realiza en el FrontEnd o donde se ejecute la aplicación

Para la visualización en los nodos esclavos se requiere aumentar una librería propia de OpenSceneGraph llamada “*osgcluster*”, la cual permite establecer una mejor comunicación entre ellos y además cuenta con un algoritmo propio para la visualización como: nodo maestro (*master*) y nodo esclavo (*slave*). De ésta manera se realizó la visualización en los nodos que se deseen, siempre habiendo un nodo maestro en el cual se ejecute la aplicación.

Programación en paralelo para los diferentes nodos del Clúster

El aplicativo con OSG pudo ser visualizado en los nodos esclavos pero luego de las pruebas realizadas se comprobó que el aplicativo no se paralelizó en los nodos de visualización, sino más bien, al ejecutar en uno de los nodos esclavos este solo distribuía el aplicativo al resto de nodos de visualización por lo que no cumple con el objetivo previsto de programación en paralelo para los diferentes nodos del Clúster, por tal motivo se requiere seleccionar otra herramienta de programación junto a una IDE que ofrezca la flexibilidad para continuar con la investigación y avanzar con los procesos de paralelización del aplicativo 3D.

Luego de observar las ventajas que proporciona la herramienta HIPerWorks para la paralelización de aplicativos con CGLX e imágenes 3D con OSGViewer se procede a realizar la elaboración de la programación en C++ con la librería CGLX para un nuevo aplicativo 3D que permita cumplir con uno de los objetivos planteados en este documento, debido a la flexibilidad que tiene esta herramienta con OSG se procede a realizar la programación y con el tiempo se acople al aplicativo OSG anterior.

Prototipo 3. Diseño e implementación del aplicativo con CGLX

Como en el prototipo anterior no se observó un avance significativo en el desarrollo del proyecto para ser paralelizado en los diferentes nodos del Clúster, se hizo necesario implementar un módulo para el manejo de los “.obj” construido en C++

junto a la librería OpenGL, Glut y CGLX siendo librerías de bajo nivel para lograr la paralelización.

El meta roll de HIPerWorks incluye todos los componentes para configurar y manejar el Clúster de visualización por lo que se deberá instalar el viz-light roll y el HIPerWorks roll en este.

El Clúster de visualización en conjunto con HIPerWorks y CGLX permitirá visualizar la paralelización del aplicativo 3D.

Características

- La herramienta HIPerWorks proporciona la configuración y gestión de los recursos del Clúster, lo que permite el funcionamiento de entornos de visualización distribuidos, la ejecución y control de aplicaciones.
- HIPerWorks es una colección de aplicaciones de productividad y de visualización interactiva de grandes colecciones de imágenes de alta resolución y renderizado 3D.
- El CGLX Core Engine API proporciona renderizado distribuido paralelizado de aplicaciones OpenGL con acceso a todas las extensiones OpenGL que se admiten a través del hardware de gráficos. El marco HIPerWorks está disponible gratuitamente para uso académico y no comercial[77].
- OSGViewer es un sistema flexible, transparente y extensible. Se lo considera como un marco escalable para entornos distribuidos y visualización de alto rendimiento para aplicaciones en OpenGL.

CGLX requiere conocer cuántos nodos esclavos están disponibles en el Clúster de visualización y cómo las pantallas se disponen en cada nodo. Se necesita crear al menos un archivo de configuración al clúster, de esta manera el sistema CGLX lo utilizará para ejecutar la aplicación en el “*wal*” (muro).

Creación de una configuración de pantalla del sistema de visualización de muro (tiled).[78]

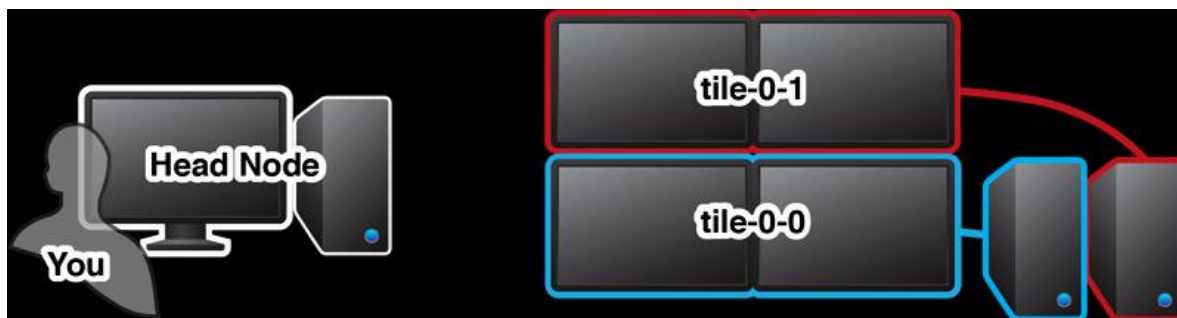


Figura 3-66 Diseño de pantalla de tiled[78]

Fuente: HIPerWorks, Quick Start, 2012

La interfaz de configuración. *Pirconfig* se compone de varios sub-módulos, permite a los usuarios efectuar una configuración dinámica y utiliza un sistema de visualización tipo muro, toma control de los nodos y distribuye las imágenes.

La barra de herramientas de HIPerWorks. Permite guardar y cargar la configuración, añadir y eliminar nodos de visualización como muestra la Figura 3-67.

El grupo de servidores. Muestra cuántos están conectados en la configuración actual. También sirve como interfaz para configurar o reconfigurar un servidor como indica la Figura 3-67.

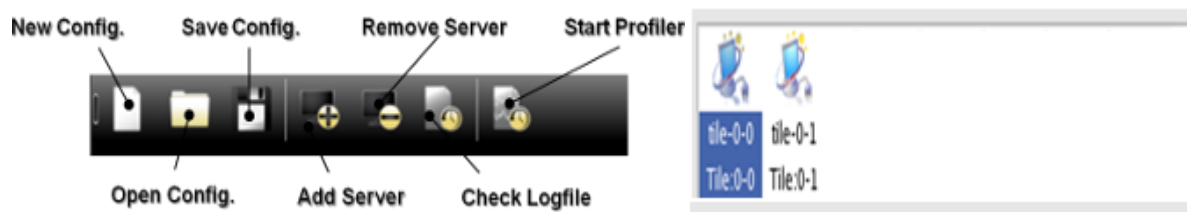


Figura 3-67 Barra de Herramientas y Grupo de servidores

Fuente: HIPerWorks, Quick Start, 2012

Cuadro de diálogo agregar servidor. Permite añadir un nuevo tile a su configuración. En el cuadro de diálogo se puede observar que espera un nombre o dirección IP. La configuración del puerto y *Appl*, para la comunicación con el demonio son configuraciones por defecto.

Cuadro de diálogo configuración del servidor. Muestra todas las pantallas disponibles en el servidor en una tabla editable, así como su resolución actual. También permite especificar las opciones para habilitarlo.

Implementación del aplicativo con CGLX

Se procede a la implementación de un aplicativo para la manipulación de los “.obj”, el cual será construido en C++ junto a las librerías OpenGL, GLUT, GLM y CGLX adecuadas para la creación y manipulación de entornos 3D.

C. Casos de uso con notación UML.

A continuación se muestra el diagrama de uso general del sistema con CGLX

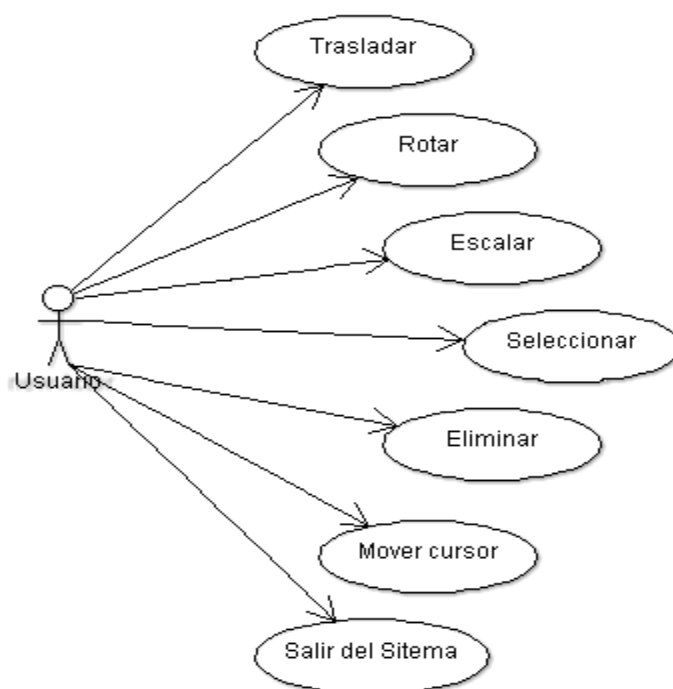


Figura 3-68 Diagrama casos de uso general del sistema con CGLX

Fuente: Los autores

Actor: Usuario

Actividad: Realiza el proceso de trasladar, rotar, escalar, seleccionar, eliminar el objeto 3D y usa los eventos del mouse en la Aplicación.

Observaciones: Al ingresar a la escena se cargará de forma automática el objeto en su respectiva capa, es decir, la capa piel.

Diseño

Iteración 1. Diseño e implementación del aplicativo con OpenGL

A. Diseño y diagrama de clases

El diseño se basa en la orientación a objetos empleados en C++, se ha estructurado la aplicación en diferentes métodos dependiendo de las distintas interacciones que se realizarán para el desarrollo del aplicativo.

Se estructura la solución de la siguiente manera, en dos Clases simples:

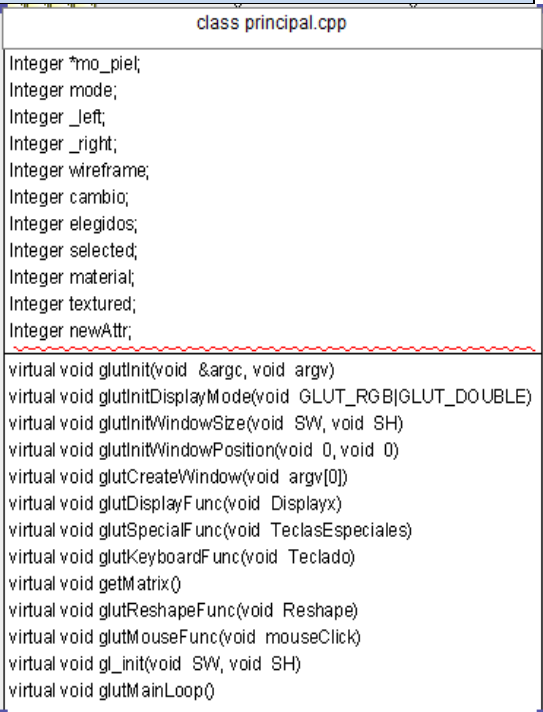
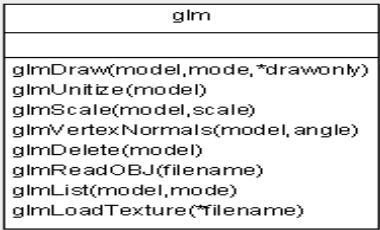
Diagrama de clases para el aplicativo con CGLX		
Clase	Función	Diagrama
Principal	Se encuentra el método <i>main</i> que ejecuta la aplicación en CGLX, contiene la estructura de los métodos para la visualización de la escena gráfica, uso de los periféricos y desde donde se llama a cargar los modelos Wavefront.	 <pre> class principal.cpp Integer *mo_piel; Integer mode; Integer _left; Integer _right; Integer wireframe; Integer cambio; Integer elegidos; Integer selected; Integer material; Integer textured; Integer newAttr; virtual void glutInit(void &argc, void argv) virtual void glutInitDisplayMode(void GLUT_RGB GLUT_DOUBLE) virtual void glutInitWindowSize(void SW, void SH) virtual void glutInitWindowPosition(void 0, void 0) virtual void glutCreateWindow(void argv[0]) virtual void glutDisplayFunc(void Displayx) virtual void glutSpecialFunc(void TeclasEspeciales) virtual void glutKeyboardFunc(void Teclado) virtual void getMatrix() virtual void glutReshapeFunc(void Reshape) virtual void glutMouseFunc(void mouseClick) virtual void gl_init(void SW, void SH) virtual void glutMainLoop() </pre>
GLM	Es una clase de carga, configuración y edición de objetos .obj desde un archivo.	 <pre> glm glmDraw(model,mode,*drawonty) glmUnitize(model) glmScale(model,scale) glmVertexNormals(model,angle) glmDelete(model) glmReadOBJ(filename) glmList(model,mode) glmLoadTexture(*filename) </pre>

Tabla 3-27 Lista de clases CGLX

Fuente: Autores

A continuación se detalla los principales métodos y funciones de la clase Principal.cpp:

Métodos y funciones de la clase principal		
Método	Función	Diagrama
Main	Se ejecuta la aplicación y permite llamar al resto de métodos para la visualización de la escena gráfica, uso de los periféricos.	<pre> classDiagram class main { *mo_piel : Integer mode : Integer _left : Integer _right : Integer wireframe : Integer cambio : Integer elegidos : Integer selected : Integer material : Integer textured : Integer newAttr : Integer glutInit(&argc,argv) glutInitDisplayMode(GLUT_RGB GLUT_DOUBLE) glutInitWindowSize(SW,SH) glutInitWindowPosition(0,0) glutCreateWindow(argv[0]) glutDisplayFunc(Displayx) glutSpecialFunc(TeclasEspeciales) glutKeyboardFunc(Teclado) getMatrix() glutReshapeFunc(Reshape) glutMouseFunc(mouseClick) gl_init(SW,SH) glutMainLoop() } </pre>
Carga	Método con el cual se realiza el llamado al modelo Wavefront y en donde se realiza la visualización del modelo como tal.	<pre> classDiagram class Cargamodelo { void Cargamodelo() } class Cargaobj { void Cargaobj(GLMmodel *pmodelo, GLuint mode, int mod) } Cargamodelo < -- Cargaobj </pre>
Select	Verifica si el objeto fue pinchado y selecciona el correspondiente modelo.	<pre> classDiagram class gl_select { void gl_select(int x, int y) } class list_hits { void list_hits(GLint hits, GLuint *names) } gl_select <--> list_hits </pre>

Métodos y funciones de la clase principal		
Método	Función	Diagrama
Mouse	Método de selección de objeto por pick y llamado a funciones de carga de modelos.	<pre> classDiagram class Principal { void mouseClicked(int button, int state, int x, int y) void mousedw(int x, int y, int but) } Principal --> Principal : mouseClicked </pre>
Teclado	Se ejecuta el traslado, escalado y rotación del modelo 3D mediante el uso de las diferentes teclas.	<pre> classDiagram class Principal { void Teclado(unsigned char key, int x, int y) getMatrix() glutPostRedisplay() } Principal --> Principal : Teclado Principal --> Principal : Teclado Principal --> Principal : Teclado </pre>
Elimina	Método que se encarga de eliminar el modelo y cargar la nueva escena gráfica sin el modelo borrado.	<pre> classDiagram class Principal { void Eliminaobj(GLuint nom) } Principal --> Principal : Eliminaobj </pre>

Tabla 3-23 Lista de métodos y funciones principal.cpp

Fuente: Autores

El Diagrama de clases para el aplicativo 3D realizado con la librería CGLX tiene la clase principal la cual realiza la ejecución de la aplicación y permite llamar al resto de métodos para la visualización de la escena gráfica y el uso de los periféricos (ver anexo 5.5.2.8).

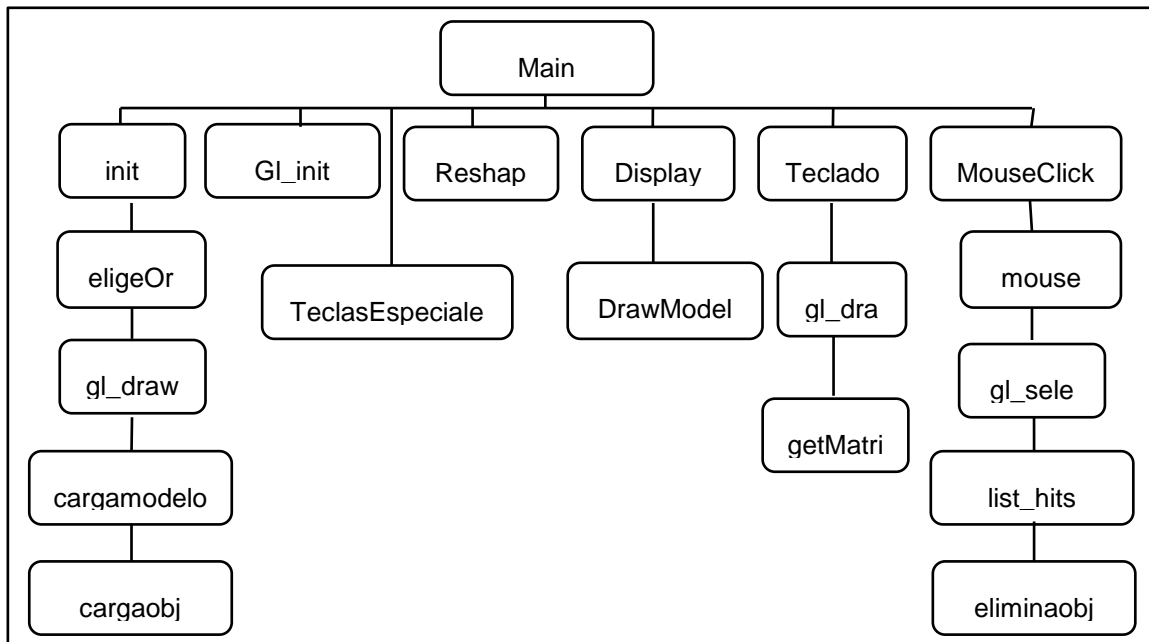


Figura 3-69 Diseño de Clases OSG.

Fuente: Autores

B. Descripción de librerías

Para iniciar con la programación 3D se debe realizar enlaces a las librerías de GL, GLUT que servirán para elaborar el aplicativo. Con estas se podrá visualizar un objeto, su color y coordenadas en las que aparecerá, así como la utilización de teclado.

Con la librería glut.h se garantiza que gl.h y glu.h estén correctamente incluidos.

```

#include<stdlib.h>
#include<memory.h>
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<unistd.h>
#include<GL/gl.h>
#include<GL/glut.h>
  
```

También se efectuará la lectura de ficheros de escena 3D con la librería GLM que servirá para cargar los archivos .obj con las texturas:

```
#include"glm/glm.h"
```

La visualización en pantalla de los objetos se lo implementará con la librería CGLX para su lectura y manipulación que será referenciada como se muestra a continuación:

```
#include<pirCore/cglX.h>
```

Codificación

Para proceder con la ejecución de un archivo CGLX con HIPerWorks se tiene que tener instalados los paquetes necesarios para su compilación (ver Anexo Instalación de paquetes en el nodo).

Iteración 1. Configuración del Wall

Obtener la licencia y autenticarla para trabajar con HIPerWorks[78]

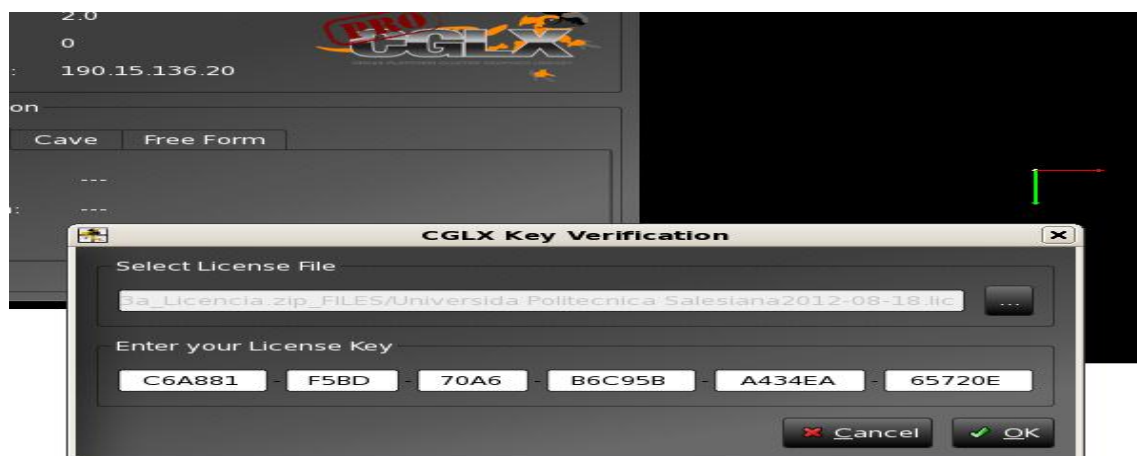


Figura 3-70 Autenticación de licencia

Fuente: Los autores

Para el enlace con los nodos de visualización se requiere proporcionar la dirección IP de red local de la Cabeza del nodo (Head) en las opciones de Preferencias:



Figura 3-71 IP del nodo Head

Fuente: Los autores

- Para iniciar un programa demonio en cada nodo del clúster tipear


```
$ pirdstart
```

- En un terminal *ssh* para cada nodo *render*, ejecutar *pirdaemon*.

```
$ ssh tile-0-1
$ pirdaemon
```

- Para iniciar HlPerWorks tipear

```
$ pirconfig
```

- Añadir el Nodo de visualización seleccionando en la barra de herramientas *add server* como indica la Figura 3-67 y luego colocar el nombre del nodo ver Figura 3-72
- Doble clic en el Nodo de visualización , esta muestra un cuadro de diálogo con una tabla editable (ver Figura 3-72) en la cual se creará el diseño de pantalla de cómo será visualizada la imagen 3D en los Nodos, si se tiene más de un nodo se procederá a incrementar el valor de la columna o fila para definir la pantalla:

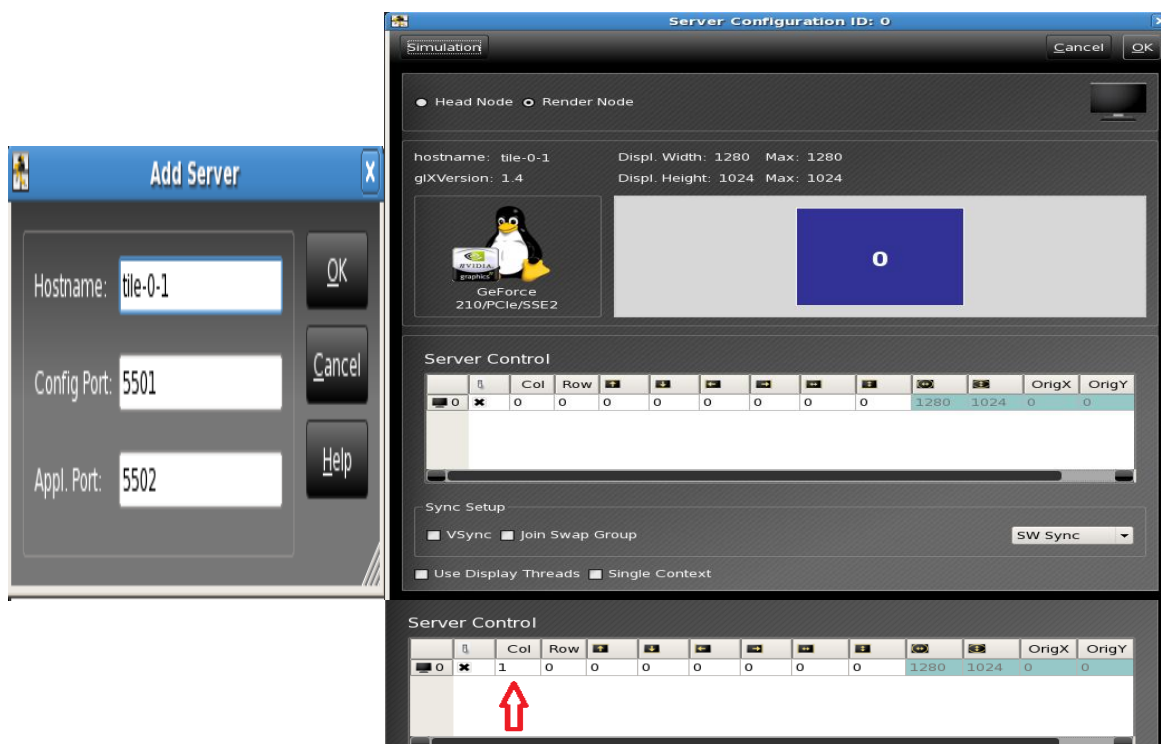


Figura 3-72 Nombre del nodo y Cuadro de diálogo con la tabla para crear el diseño de pantalla para los nodos

Fuente: Los autores


- Visualizar una vista 3D de los monitores agregados y un icono de monitor  , indicando que el servidor se ha modificado:



Figura 3-73 Vista 3D de los monitores agregados

Fuente: Los autores

- Guardar la configuración e ingresar como nombre de archivo *csdefault.xml*.
- Para ejecutar, buscar el aplicativo y dar clic en *start*.
- Para poder observar con el visualizador de HIPerWorks se procede a dar clic en *select program*, el cual desplegará una ventana donde se buscará y seleccionará el archivo CGLX, luego dar clic en *open* y *start*.

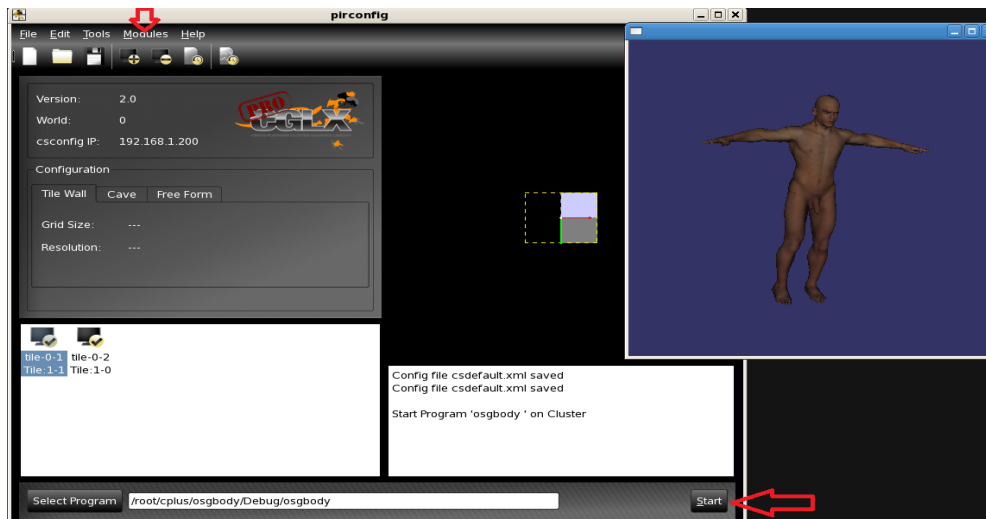


Figura 3-74 Vista 3D de los monitores agregados

Fuente: Los autores

Iteración 2. Diseño e implementación del aplicativo con CGLX

Capa: Principal

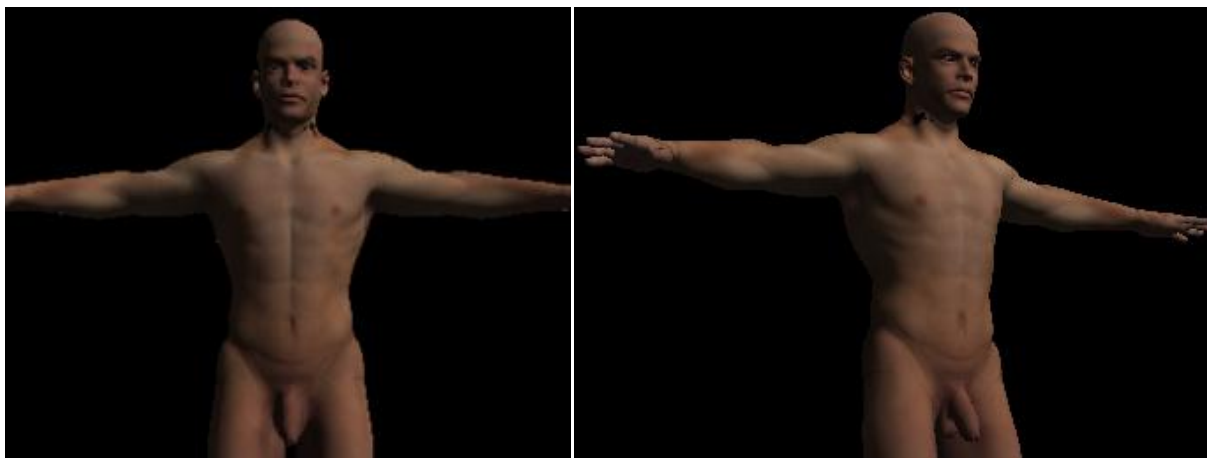


Figura 3-75 Ventana de la capa piel

Fuente: Autores

Módulo: Capa principal

Pantalla: Aplicativo capa principal

Proceso: Ejecuta el inicio del sistema para la creación de la escena.

Evento: Selección en el menú con el puntero del mouse y clic

Descripción: La clase principal será la que inicialice el sistema mediante la creación de la escena. Permite unir y enlazar todos los métodos, clases y librerías al *main*. Contiene el método que llama a las distintas capas del cuerpo humano siendo la capa piel la primera en ser visualizada y retirada al seleccionarla.

Parte del código se describe a continuación:

Como primera instancia se crearán variables para el control de la vista, del mouse, del teclado, de matrices, la carga de materiales, la carga luz (*light*), la carga de modelos:

```
GLMmodel *pmodel = NULL;
int _mouseX = 0;
double _matrix[16];
int material = 1;
```

Se especificará el tipo de iluminación suave o *SMOOTH* que dará una calidad considerable con un coste computacional que no requiera de mucho proceso.

```
#ifdef SMOOTH_HINT
int smooth_hint = 0;
#endif
```

Cargamodelo() llama a los archivos .obj que contienen las partes del cuerpo humano.

Se limpiará el *frame buffer* con el color de *Clear* (negro), también se especificarán los parámetros en cuanto a material, se activarán las luces y se utilizará el *buffer* de profundidad para la eliminación de caras ocultas.

```
void Cargamodelo (){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);

    ...

    switch (cambio) {
    case 1: { // Piel
        glLoadName(1);
        pmodel = glmReadOBJ("data/bodypiel_org.obj"); //
        break;
    }
    case 2: { // Músculos
        glLoadName(1);
        pmodel = glmReadOBJ("data/bodymusculo1.obj"); //
        glTranslatef(0, 0, 0);

        ...

        break; }

        ...

    default: { printf("\n No hay opción");
        break; }
    }

    if (!pmodel) { printf("\n Error ");
        exit(0); }

    glLoadName(1);
    glmUnitize(pmodel); /// Modeloentamaño Standard
    glmVertexNormals(pmodel, 90.0, GL_TRUE);
```



```

    glmDraw(pmodel, mode);
    ...
    glPopMatrix();
    glutPostRedisplay();    }

```

En la clase *main* se creará una ventana para visualizar a los objetos en la escena a la cual se le otorgará un nombre, también se llamará a la función que realice el redibujado, por programa o por redimensión de la ventana y las funciones para el control del mouse y del teclado.

```

Int main(int argc, char **argv){
    cgIInit(&argc, argv);
    cgIInitWindowSize(1024, 768);
    cgICreateWindow("CuerpoHumano 3D");
    cgIDisplayFunc(Displayx); // cada vez que ocurra un cambio en la ventana se
    ejecuta la función dibujar
    cgIKeyboardFunc(Keyboard); //función de control eventos con el teclado
    cgISpecialFunc(TeclasEspeciales);
    cgIReshapeFunc(Reshape); // Función de control del cambio de tamaño de la
    ventana de visualización
    cgIMouseFunc(Mouse); // Función de control de eventos con el ratón
    cgIMotionFunc(Motion);

```

Se definirá el color de fondo para la ventana cada vez que se vuelva a dibujar y se activarán las fuentes de luz o los cálculos en general de iluminación en el lugar determinado.

```

glClearColor(0.0, 0.0, 0.0, 0.0);
glClearAccum(0.0, 0.0, 0.0, 0.0);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

```

```

glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glDepthFunc(GL_LESS);
glEnable(GL_DEPTH_TEST);
glEnable(GL_NORMALIZE);
glEnable(GL_TEXTURE_2D);

```

Se cargará el primer objeto con un modelo de tamaño estándar.

```

pmodel = glmReadOBJ("data/bodypiel_org.obj"); /// Cargade OBJ
glmUnitize(pmodel); /// Modeloentamaño Standard
glmVertexNormals(pmodel, 90.0, GL_TRUE);
glutMainLoop();
exit(0); } }

```

reshape() este método controla el cambio de tamaño de la ventana de visualización tras ser redimensionada.

```

void Reshape(int w, int h){
glViewport(0, 0, w, h);
...
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

```

```
gluPerspective(45, (double) w / (double) h, 0.1, -_zFar);
glMatrixMode(GL_MODELVIEW); }
```

Mouse() este método controla los eventos del ratón cada vez que se de clic.

```
void Mouse(int button, int state, int x, int y){
if (state == GLUT_UP)
    switch (button) {
    case GLUT_LEFT_BUTTON:
        _mouseLeft = false;
        glmDelete(pmodel);
        Displayx();
    break; ...
    } else
    switch (button) {
    case GLUT_LEFT_BUTTON:
        _mouseLeft = true;
    break; ...
    }
glGetIntegerv(GL_VIEWPORT, viewport);
getMatrix();
glutPostRedisplay();//se le dice a OpenGL que dibuje de nuevo cuando pueda
}
```

Eliminaobj() permite la eliminación de objetos, reconoce el objeto seleccionado y lo elimina de la aplicación, refresca el entorno y construye un nuevo modelo sin el objeto borrado:

```
void Eliminaobj(GLuint nom) {
if(nom==1) {
```

```

glmDelete(pmodel);
gl_draw();
printf(" Accion - Borrado pmodel\n");
cambio++;
Cargamodelo();
} }

```

Keyboard() permite la manipulación de los objetos con los eventos del teclado:

Para aplicar la rotación se utilizará *glRotatef()* que tiene cuatro argumentos: el primero es el ángulo que se desea girar en grados sexagesimales en sentido contrario a las agujas del reloj; los siguientes tres argumentos definen las coordenadas x, y, z del eje alrededor del cual se realizará el giro. Con la función *glScalef()* se cambiará el tamaño de los objetos, este posee tres argumentos que son los factores de escala para cada uno de los tres ejes, de manera que si estos valores no son iguales, además de cambiar el tamaño, se realiza una deformación del objeto. Los factores mayores de 1 aumentan las coordenadas y los menores de 1 las disminuyen, la siguiente tabla muestra el detalle que hará cada tecla al ser presionada.

Eventos del teclado	
Evento	Detalle
KEY_O	Vuelve el grupo de objetos al origen en la posición 0
KEY_D	Rotación en el eje X positivo
KEY_A	Rotación en el eje X negativo
KEY_E	Realiza el aumento de zoom en el eje Y
KEY_Q	Realiza la disminución de zoom en el eje Y
KEY_Left	Traslada en forma horizontal en el eje negativo
KEY_Right	Traslada en forma horizontal en el eje positivo
KEY_Down	Traslada en forma vertical en el eje negativo
KEY_Up	Traslada en forma vertical en el eje positivo

Tabla 3-28 Eventos del teclado para manipulación de los objetos 3D

Fuente: Autores

```

void Keyboard(unsigned char key, int x, int y){
switch (key) {
    case'e':{double s = exp((double) 5 * 0.01);
                glScalef(s, s, s);
            break;}
    case'q':{double s = exp((double) -5 * 0.01);
                glScalef(s, s, s);
            break;}
    case'a':{
                glRotatef(5, 0, 0, 1);
            break;}
}
}

```

```

    case'd':{
                glRotatef(-5, 0, 0, 1);
            break;}
    case'o':{ ...//resetea la escena
            break;}
    ...
break;}
case 27: { ... break; }
default:{ fflush(stdin);break; } }
fflush(stdin);
getMatrix();
glutPostRedisplay(); }

```

TeclasEspeciales() permite la manipulación de los objetos con los eventos del teclado los cuales se utilizarán para la traslación en el eje de coordenadas x,y:

```

void TeclasEspeciales(int a_keys, int x, int y){ ...
glGetIntegerv(GL_VIEWPORT, viewport);
switch (a_keys) {
    case GLUT_KEY_UP:{
        glLoadIdentity();
        glTranslatef(0, 0, 0.05); //0.05 Se realiza una traslación en la coordenada y +
        glMultMatrixd(_matrix);
        break; }
        case GLUT_KEY_DOWN:{ ...
// glTranslatef(0, 0, -0.05); realiza una traslación en la coordenada y-
        break; }
        case GLUT_KEY_RIGHT:{ ...
// glTranslatef(0.05, 0, 0); realiza una traslación en la coordenada x+
        break; }
    case GLUT_KEY_LEFT:{...
// glTranslatef(-0.05, 0, 0); realiza una traslación en la coordenada x-
        break; }
    default:
        fflush(stdin);
        break; }
        fflush(stdin);
        getMatrix();
        glutPostRedisplay();
} }

```

La función DrawModel() es el modo de dibujo para que los objetos sean cargados en la escena

```

void DrawModel(void) {
mode = GLM_NONE;      // Modo dereseteo
if (smooth)    mode = mode | GLM_SMOOTH;
elsemode = mode | GLM_FLAT;
if (two_sided)mode = mode | GLM_2_SIDED;
if (material)mode = mode | GLM_MATERIAL;
elsemode = mode | GLM_COLOR;
if (textured && material)    mode = mode | GLM_TEXTURE;
glPushMatrix();
if(cambio==2){
if (pmodel)    {
    glmDraw(pmodel, mode);
    glmDraw(pmodel1, mode);
    glmDraw(pmodel2, mode);
    }}
else{
if (pmodel)    {glmDraw(pmodel, mode);  }
}
glPopMatrix();
}

```

Función que se encarga de realizar el dibujo en una pantalla cada vez que se redimensiona la ventana.

```

void Displayx(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glLoadIdentity();
    glTranslatef(0, 0, centerZ);

```

```

glRotatef(-90, 1, 0, 0);
glMultMatrixd(_matrix);
if (wireframe) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
else
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
DrawModel();
glTranslatef(0, 0, -centerZ);
glPopMatrix();
glutSwapBuffers();
}

```

El proceso para ejecutar el archivo CGLX se lo puede revisar en Anexo Proceso de compilación con CGLX

Pruebas del aplicativo 3D con HIPerWorks

Para la visualización de imágenes con .osg que fue guardada con la aplicación OSG anterior se procede a dar clic en *modules/OSGViewer*, el cual desplegará una ventana donde se buscará y seleccionará el archivo con la extensión, luego dar clic en *open* y *start*, siendo posteriormente visualizadas en los nodos (ver Figura 3-76):

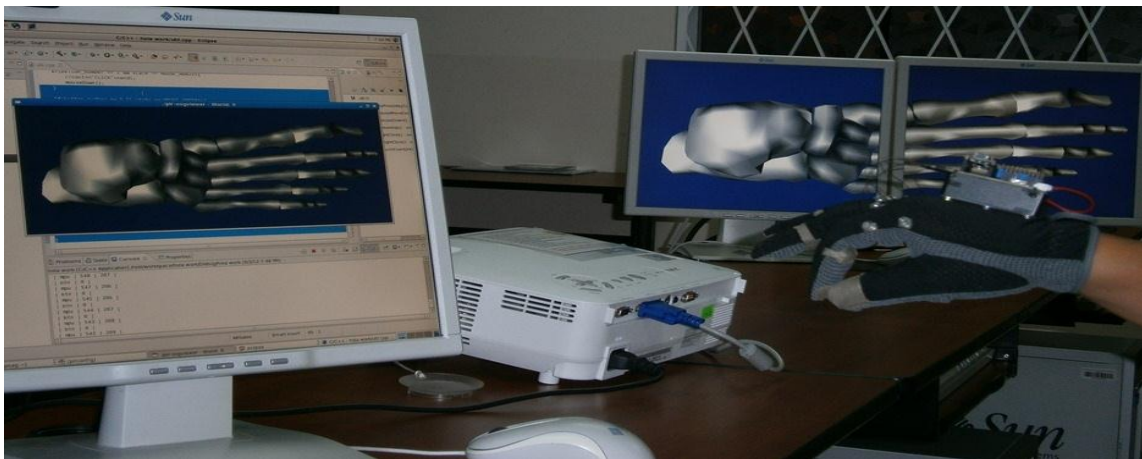


Figura 3-76 Imagen .osg visualizada en el FrontEnd y en los nodos junto al dispositivo electrónico

Fuente: Los autores

Para la visualización paralela del aplicativo 3D con CGLX proceder con los pasos de la Iteración 1. Configuración del Wall y del Anexo Proceso de compilación con CGLX, ejecutar el archivo e iniciar con la manipulación de los objetos comenzando desde la capa piel, la capa músculos, la capa órganos, el sistema respiratorio y los huesos dando como resultado la siguiente imagen:

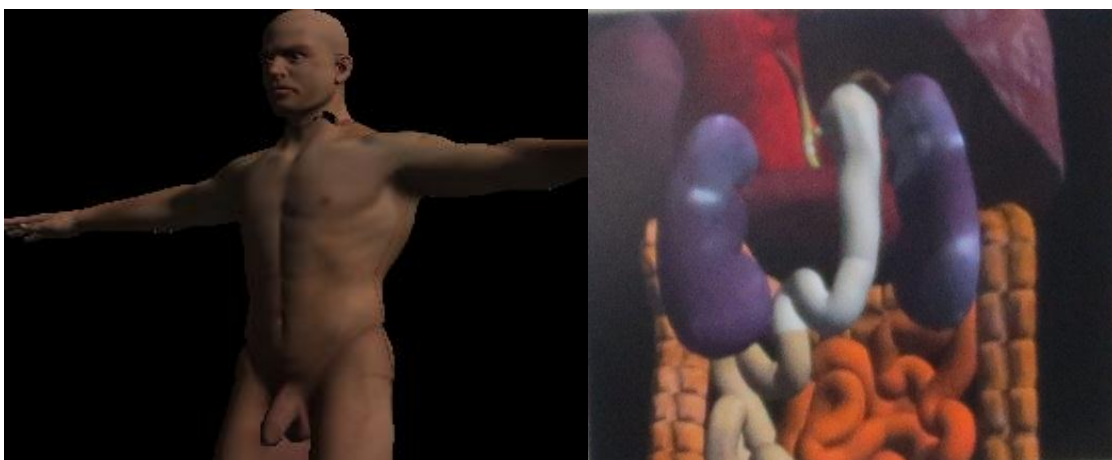


Figura 3-77 Ventana de aplicativo con CGLX del cuerpo humano

Fuente: Autores

Comprobación de errores

Error 01: El HIPerWorks no ejecuta el aplicativo con OSG solo visualiza imágenes.

El HIPerWorks ejecuta en paralelo solo imágenes con el osgviewer (visualizador de imágenes de OSG) y al intentar ejecutar el aplicativo realizado con OSG genera un error de no encontrar instaladas las librerías de OSG.

Error 02: Al instalar el rollHIPerWorks en los nodosvirtuales estos no se cargan completamente.

Cuando la instalación del rollHIPerWorks se completa, la máquina reinicia para hacer un reconocimiento de la tarjeta gráfica Nvidia y dado que esta no posee la tarjeta gráfica genera un error (ver figura 4.13) por lo que no se puede visualizar en los nodos virtuales.

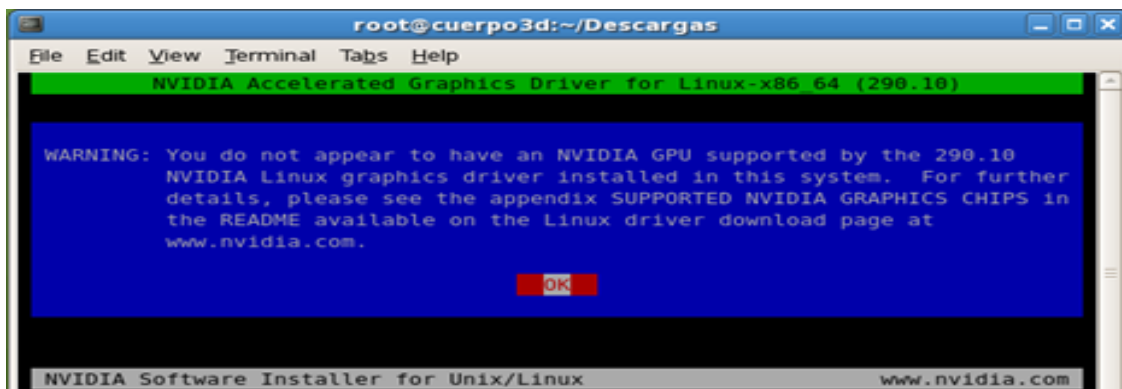


Figura 3-78 Mensaje ocurrido al intentar actualizar la tarjeta gráfica Nvidia

Fuente: Los autores

Error 03: Las imágenes de las texturas de los objetos no se visualizan en el aplicativo CGLX

En el Archivo “MTL” (.mtl) (ver Figura 3-79) se encuentra la declaración *map_Kd* que contiene el nombre del archivo de imagen con su dirección, al ejecutar el aplicativo con OSG este no es leído y no coloca la textura dando como resultado la imagen en gris.

```
# 3ds Max Wavefront OBJ Exporter v0.97b - (c)2007 guruware
newmtl 01___Default
  Ns 9.999999046326
  Ni 1.500000000000
  d 1.000000000000
  Tf 1.000000000000 1.000000000000 1.000000000000
  illum 2
  Ka 0.588235318661 0.588235318661 0.588235318661
  Kd 0.588235318661 0.588235318661 0.588235318661
  Ks 0.000000000000 0.000000000000 0.000000000000
  Ke 0.000000000000 0.000000000000 0.000000000000
  map_Kd /Model_Skin/cara.jpg
```

Figura 3-79 Archivo “MTL” (.mtl) del archivo de imagen Model_Skin/cara.mtl

Fuente: Autores

Error 04: Todas las capas no pueden ser cargadas al mismo tiempo.

Al momento de intentar cargar las capas junto a los músculos estos no permiten la ejecución del aplicativo debido a que la librería GLM limita el archivo a cargar.

Error 05: Existen conflictos con la librería GLM al momento de cargar los objetos.

La biblioteca GLM no soporta la lectura de la posición dada desde 3D Max y por ello crea conflictos al momento de cargar los objetos con las nuevas coordenadas establecidas desde el aplicativo.

Solución de errores

Solución 01: El HIPerWorks no ejecuta el aplicativo con OSG solo visualiza imágenes

Se procede a realizar la instalación del OSG en el Clúster (ver anexo B 1.1), al volver a ejecutar el aplicativo con OSG ya es visualizado por el FrontEnd e interactúa con el teclado y mouse.

Solución 02: Al instalar el roll HIPerWorks en los nodos virtuales estos no se cargan completamente.

Para visualizarlos objetos 3D los nodos deben ser instalados físicamente y se requiere de la adquisición de una tarjeta gráfica Nvidia ya que el roll HIPerWorks solo trabaja con esta tarjeta.

Solución 03: Las imágenes de las texturas de los objetos no se visualizan en el aplicativo CGLX

La carpeta en la cual se ejecuta el aplicativo deberá contener los archivos de las imágenes (.mtl, .jpg) y en la declaración *map_Kd* se deberá borrar la dirección antepuesto por el “*slash*” dejando el nombre del archivo de imagen por ejemplo: “*map_Kd Model_Skin/cara.jpg*”, con esto será posible visualizar la textura en el escenario gráfico.

Solución 04: Todas las capas no pueden ser cargadas al mismo tiempo

Se procede a cargar por partes el aplicativo, es decir, la primera capa en ejecutar será solo la capa piel, luego al ser retirada se dibujará la capa músculos y un par de órganos para que pueda ser generada, luego aparecerá el resto de órganos y los huesos.

Distribución del trabajo en los nodos de visualización

En las siguientes tablas se muestran los reportes que se generan en el FrontEnd y en los nodos de visualización al ejecutar el aplicativo en Ganglia, dependiendo del tipo de servidor y sus características.

La Tabla 3-29 muestra que al ejecutar el aplicativo como FrontEnd en la máquina física el tiempo de ejecución es mayor que al ejecutar en la máquina virtual que posee mayor memoria y procesadores, la carga se evidencia solo en el FrontEnd y en los nodos de visualización la carga es mínima debido a que son imágenes 3D, ya que con HIPerWorks se evidencia carga si son archivos grandes, por ejemplo si fuese un video de más de 1GB de tamaño visualizado con la carga seria evidente.

En la Tabla 3-30 se evidencia que en el FrontEnd de la máquina virtual realiza trabajo y en uno de los nodos de visualización físicos por un instante al momento de manipular el aplicativo cuando se realizan varias acciones.

Tabla 3-31 muestra que solo el FrontEnd de la máquina virtual realiza trabajo pero ya no se evidencia en ninguno de los nodos.

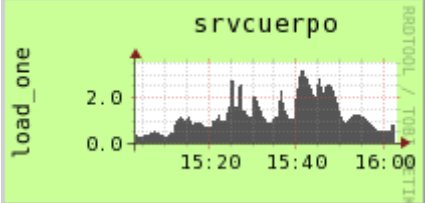
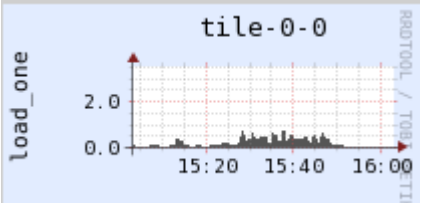
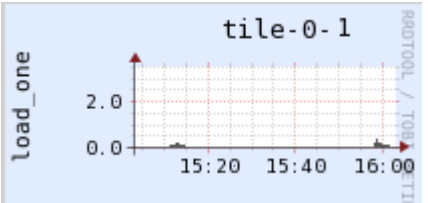
Reportes que se generan en el FrontEnd y en los nodos de visualización						
Tipo Computador	Procesador	Tarjeta Gráfica	RAM	Sistema Operativo	Tiempo de ejecución	Reporte Clúster (# Nodos)
Servidor físico (local)	Dual Core AMD Opteron(tm) processor 1218 (2 CORE)	Nvidia GeForce 8400 (1GB)	3,00 GB	CentOS 5 (64-bit)	3.10 min	
Nodos Físicos	Dual Core AMD Opteron(tm) processor 1218 (2 CORE)	Nvidia GeForce 8400 (1GB)	3,00 GB	CentOS 5 (64-bit)	2.10 min	<p>Nodo 1</p> 
						<p>Nodo 2</p> 

Tabla 3-29 Distribución del trabajo en los nodos de visualización

Fuente: Autores

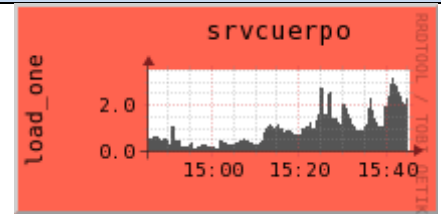
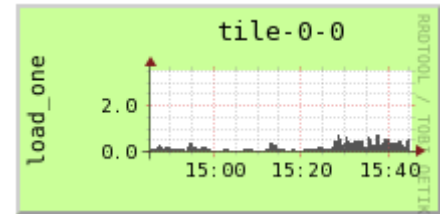
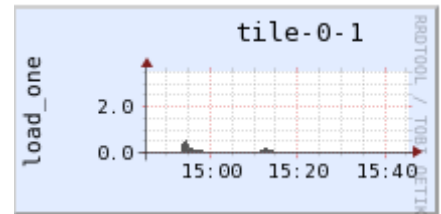
Reportes que se generan en el FrontEnd y en los nodos de visualización						
Tipo Computador	Procesador	Tarjeta Gráfica	RAM	Sistema Operativo	Tiempo de ejecución	Reporte Clúster (# Nodos)
Servidor Virtual	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz (4 CORE)	--	8,00 GB	CentOS (64-bit) 5	1.10 min	
Nodos Físicos	Dual Core AMD Opteron(tm) processor 1218 (2 CORE)	Nvidia GeForce 8400 (1GB)	8,00 GB	CentOS (64-bit) 5	1.10 min	Nodo 1
						
						Nodo 2
						

Tabla 3-30 Distribución del trabajo en los nodos de visualización

Fuente: Autores

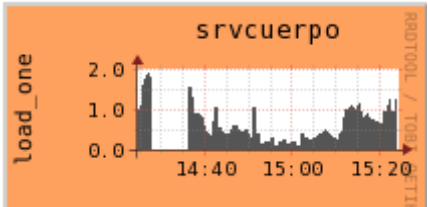
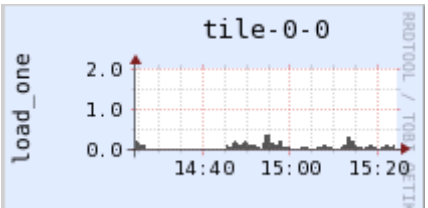
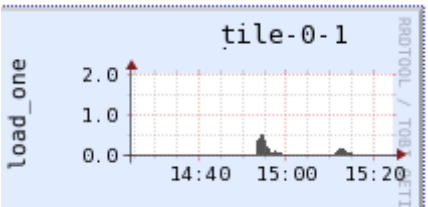
Reportes que se generan en el FrontEnd y en los nodos de visualización						
Tipo Computador	Procesador	Tarjeta Gráfica	RAM	Sistema Operativo	Tiempo de ejecución	Reporte Clúster (# Nodos)
Servidor Virtual	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz (4 CORE)	--	8,00 GB	CentOS (64-bit) 5	1.10 min	
Nodos Físicos	Dual Core AMD Opteron(tm) processor 1218 (2 CORE)	Nvidia GeForce 8400 (1GB)	8,00 GB	CentOS (64-bit) 5	1.10 min	<p>Nodo 1</p>  <p>Nodo 2</p> 

Tabla 3-31 Distribución del trabajo en los nodos de visualización

Fuente: Autores

CAPÍTULO IV

4. FUNCIONAMIENTO DEL SISTEMA

Este capítulo describe la implementación del aplicativo 3D con las imágenes “.obj”, que son manipuladas tanto con el mouse y teclado así como también con el dispositivo electrónico en el Clúster de visualización y en la página HTML.

La ejecución del aplicativo se basa en los pulsos predeterminados del dispositivo electrónico y la facilidad con que el usuario consiga interactuar con la aplicación 3D, de tal forma que puedan realizarse de manera correcta y precisa.

4.1 Prototipo 1 Implementación del Aplicativo con Java 3D en el Clúster

La Figura 4-1 muestra una pestaña de la página Web en cuyo interior se encuentran dos sub ventanas, las cuales presentan la visualización del Applet 3D del cuerpo humano estas imágenes se encuentran alojadas dentro del servidor, en la primera ventana se encuentra la capa músculos y en la otra el resto de capas a dichos objetos se los puede rotar, trasladar, zoom, seleccionar y eliminar cuya manipulación se la realiza con el mouse y el teclado.

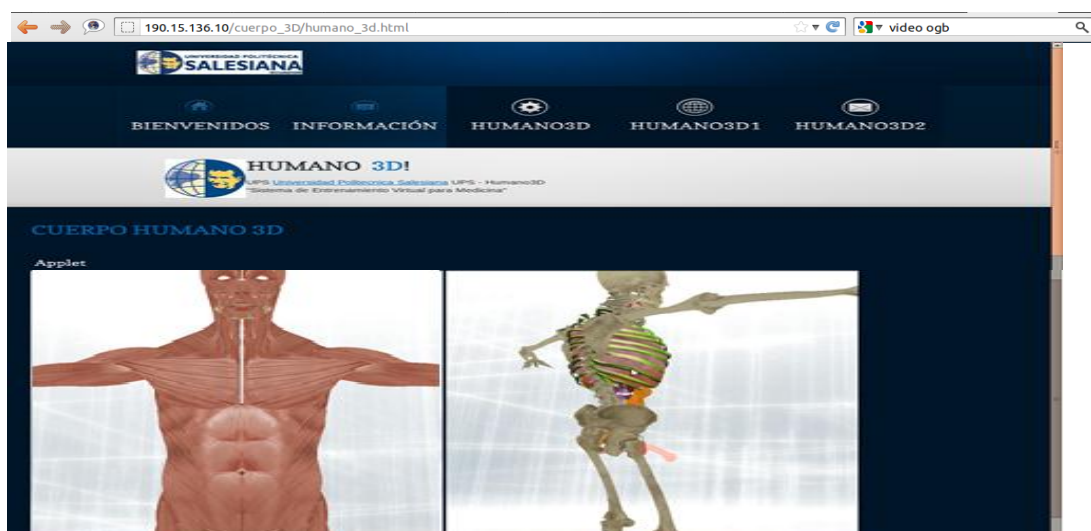


Figura 4-1 Página de visualización del Applet 3D en la Web

Fuente: Autores

En la Figura 4-2 se puede apreciar el funcionamiento de los dos módulos, muestra la iteración que realiza el dispositivo electrónico dentro del entorno 3D de la página Web, donde se realizan acciones de rotar, trasladar, zoom, seleccionar, eliminar los objetos dentro de la escena.



Figura 4-2 Ventana de iteración del dispositivo electrónico con el Applet 3D

Fuente: Autores

4.2 Prototipo 2. Implementación del aplicativo con OpenSceneGraph

El funcionamiento del aplicativo OSG distribuido en el Clúster de visualización puede ser manipulado con el teclado y mouse así como con el dispositivo electrónico realizando las acciones de rotar, trasladar, zoom, seleccionar y eliminar los objetos dentro de la escena, la Figura 4-3 muestra la interacción del dispositivo electrónico al realizar una rotación del objeto 3D.



Figura 4-3 Ventana de iteración del dispositivo electrónico con el aplicativo OSG

Fuente: Autores

La Figura 4-4 indica que al realizar un contacto del dedo pulgar con el índice realiza un pick seleccionado al objeto en la escena mostrandolo de color distinto.



Figura 4-4 Ventana de objeto seleccionado con el dispositivo electrónico

Fuente: Autores

4.3 Prototipo 3. Implementación del aplicativo con CGLX

El aplicativo CGX es ejecutado en el FrontEnd de la máquina virtual, se indica la visualización paralela en los nodos del Clúster y se realiza la manipulación de los objetos con el dispositivo electrónico como muestra la Figura 4-6.

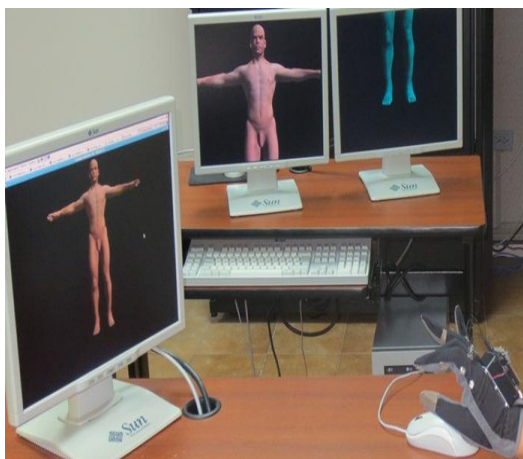


Figura 4-5 Aplicación con CGLX en el FrontEnd del Clúster y en los nodos de visualización
Fuente: Autores

La Figura 4-6 muestra como el aplicativo es manejado por el dispositivo electrónico, realizando operaciones de traslación, rotación, zoom, selección y eliminación del objeto al hacer contacto el pulgar con uno de los sensores que poseen los otros dedos.

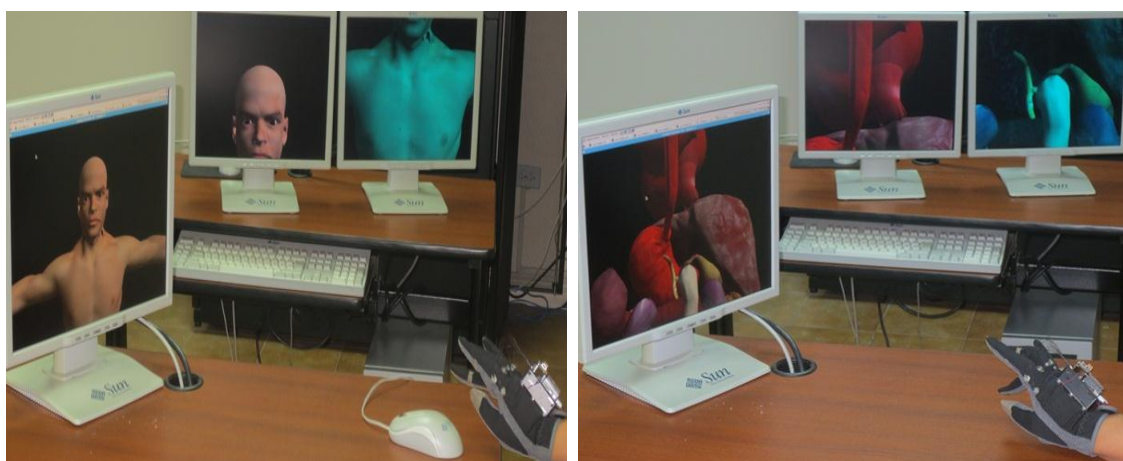


Figura 4-6 Aplicación con CGLX en el FrontEnd del Clúster y en los nodos de visualización
Fuente: Autores

CONCLUSIONES

- Este proyecto ha sido en su mayoría de tipo investigativo, inicialmente la programación y configuraciones fueron desconocidas en su mayoría, se debió aprender y analizar librerías, instalaciones, configuraciones y programación 3D. Se ha efectuado un completo análisis acerca de comunicación por hardware utilizando herramientas de software libre. También se ha realizado una investigación acerca de algoritmos de programación distribuida para procesamiento en el Clúster, el cual integra una cabeza y dos nodos de visualización, con lo cual cumple con el objetivo de paralelización visual del aplicativo.
- El aplicativo cuenta con varios módulos que se los ha desarrollado a medida para el uso de este proyecto; el módulo realizado en C++ utiliza librerías que integran los datos provenientes del dispositivo electrónico con el aplicativo OSG, el Applet 3D y el aplicativo CGLX; los módulos del aplicativo OSG, Applet 3D y aplicativo CGLX contienen al cuerpo humano, sus texturas, elementos “.obj” y algoritmos para la traslación, rotación y escalado; el módulo de página Web realizada en HTML contiene al Applet 3D el cual se encuentra en el Clúster de visualización y los módulos del aplicativo con OSG y CGLX serán distribuidos y visualizados en el mismo, con la diferencia que el aplicativo OSG distribuye la visualización a los nodos y el aplicativo CGLX lo paraleliza visualmente con la ayuda de la herramienta HPerWorks. Estos módulos actúan de manera independiente, las aplicaciones OSG, CGLX y Applet funcionan de igual manera mediante los periféricos teclado y mouse así como con el dispositivo electrónico. Se puede hacer uso del código y tal vez una futura edición, ya que es generado y editado con código abierto.
- Se ha desarrollado un software que integra un dispositivo electrónico que envía datos de tipo “String” (cadena de caracteres) por el puerto serial USB y un aplicativo que permite la traslación, escalado y rotación de imágenes 3D, además de seleccionar y eliminar capas del cuerpo humano, este aplicativo realizado con OpenSceneGraph y mediante la herramienta HPerWorks, permite el funcionamiento de entornos de visualización distribuida, la ejecución y control de la aplicación con los recursos del

Clúster, el servidor (nodo cabeza) gestiona la información y visualiza las imágenes 3D en los nodos de visualización del Clúster por medio de la Red Avanzada.

- Debido al inconveniente, de no poder paralelizar el aplicativo, se debió hacer varios cambios, entre ellos y el más importante, fue buscar otra herramienta como HIPerWorks para que paralelice el aplicativo y con ello realizar otro aplicativo en CGLX, un aplicativo de igual características y similar diseño que OSG y Java 3D, que permite trasladar, escalar y rotar, de igual manera seleccionar y eliminar los objetos 3D; con los tres aplicativos juntos se cumplen los objetivos planteados, pero que no favorecieron en terminar el proyecto a tiempo, puesto que se llevó tiempo realizar el tercer aplicativo, aunque el diseño y varios algoritmos son iguales, la programación es muy diferente en cada uno de ellos con un cierto grado de complejidad y se necesitó investigar acerca del nuevo aplicativo en 3D.
- Con la investigación realizada se conoce que existen dos tipos de paralelización: proceso de datos y de visualización; se eligió esta última con la cual se obtuvo la manera para paralelizar visualmente los objetos 3D en varios displays, los cuales están integrados a un Clúster Rocks con el viz roll HIPerWorks para funciones de visualización gráfica, permitiendo interactuar con el dispositivo electrónico y mejorando el rendimiento de los gráficos.

RECOMENDACIONES

- El proyecto requiere de un buen manejo de la herramienta 3D Max en lo referente a creación, edición de objetos y conocimiento de diseño gráfico al momento de realizar las partes del cuerpo humano uno a uno, ya que se debe ir editando los objetos básicos dados en 3D Max, graficar en 3D los órganos según el Atlas del cuerpo humano y que logre similitud a la imagen real que se está siguiendo.
- Para próximos trabajos se recomienda hacer una investigación más profunda acerca de las librerías, clases y métodos de OSG, puesto que es un tema muy amplio y tiene gran alcance en lo que a 3D se refiere.
- No es recomendable hacer uso del Applet 3D en aplicaciones que requieren imágenes de gran tamaño en píxeles (8192x8192 px) o de gran tamaño en disco (10Mb), ya que como se ha visto, al tener varias imágenes se limita la carga, visualización, manipulación y rapidez del aplicativo, más aún si estas son adheridas en una página Web.
- Para el uso del dispositivo electrónico primero leer el manual de usuario que viene adjunto previo a la utilización, de esta manera se conocerá de forma detallada su calibración y obtención de datos para la posterior manipulación de las imágenes 3D del cuerpo humano.
- Para las instalaciones de los programas que se ejecuten en Linux (CentOS o Ubuntu 11.04) se recomienda comprobar que las rutas (Path) a los archivos estén correctamente definidas, caso contrario los aplicativos que intenten ejecutarse no podrán encontrar las librerías.
- Es recomendable que los nodos sean de similares características físicas, especialmente que las tarjetas gráficas sean Nvidia, ya que este es un requerimiento para usar Hyperworks. Además al momento de instalar las bibliotecas su configuración sería similar en todos los nodos y no habría mayores inconvenientes.
- Para trabajos futuros con HIPerWorks y CGLX se recomienda leer tutoriales, ver manuales e ingresar a los foros, puesto que son librerías que proporcionan ventajas en cuanto se refiere al área de investigación, aunque requieren conocimientos de programación avanzada de C++, básica de 3D y conocimiento de configuración de Clústers.

BIBLIOGRAFÍA

- [1] “Metodologías de desarrollo del software.” [Online]. Available: <http://latecladeescape.com/articulos/1550-metodologias-de-desarrollo-del-software>. [Accessed: 17-Oct-2012].
- [2] M. P. ARTEAGA CHAMORRO and D. S. GUAMÁN LOACHAMÍN, “Desarrollo de una interfaz gráfica de usuario para la administración de dispositivos de red por medio de net-snmp en el sistema operativo Linux,” ESCUELA POLITÉCNICA NACIONAL, Quito, 2010.
- [3] Molpeceres, A., “Procesos de desarrollo: RUP, XP y FDD.” 2002.
- [4] R. A. C. Gil, “Estructura básica del proceso unificado de desarrollo de software,” *Sistemas & Telemática*, vol. 2, no. 3, Jul. 2006.
- [5] “Example FuseBox Project,” 2003. [Online]. Available: http://www.insite-media.com/internet_marketing/workzone.cfm. [Accessed: 18-Oct-2012].
- [6] “Fusebox Lifecycle Process.” [Online]. Available: <http://www.corfield.org/FLiP/index.cfm?fuseaction=methodology.steps>. [Accessed: 17-Oct-2012].
- [7] V. R. J. C. Amaro Calderón Sarah Dámaris, “Metodologías Ágiles,” Universidad Nacional de Trujillo, Trujillo – Perú, 2007.
- [8] “CyTA,” vol. 05, p. <http://www.cyta.com.ar>, Jun-2006.
- [9] M. A. Vidal Ivan and Lasso Diego, “Adaptive Software Development,” *ASD(Adaptive Software Development)*, 13-Jun-2010. [Online]. Available: <http://www.slideshare.net/urumisama/metodologia-agil-asd>. [Accessed: 16-May-2012].
- [10] Medina J. Álvaro, “Proyecto de realidad virtual ayuda a rehabilitar a niños de la Teletón,” *Nación.cl*, 27-Dec-2012. [Online]. Available: <http://www.lanacion.cl/noticias/site/artic/20121227/pags/20121227104755.html>. [Accessed: 06-Feb-2013].
- [11] Universidad Jaume, “LABPSITEC,” *Laboratorio de Psicología y tecnología*, 2012. [Online]. Available: <http://www.labpsitec.uji.es/esp/proyectos/proyectos.php>. [Accessed: 06-Feb-2013].
- [12] K. Vega, “Visualización en TACC.” 2012.
- [13] HIPerWorks, “High Performance Visualization Systems, Visualization Clusters, Tiled Display Systems - Installations,” *HIPerWorks*, 2012. [Online]. Available: <http://www.hiperworks.com/index.php/our-clients>. [Accessed: 07-Feb-2013].
- [14] CSIRO, “CSIROvision – a global collaboration platform,” 2012. [Online]. Available: <http://www.csiro.au/resources/CSIROvision>. [Accessed: 07-Feb-2013].
- [15] Fernández Víctor Fresno, “Lenguajes de marcado.” .
- [16] “Características del lenguaje HTML,” *Página Web sobre HTML y CSS*. [Online]. Available:

- <https://belenus.unirioja.es/~guprado/pagweb/carachtml.html>. [Accessed: 17-Oct-2012].
- [17] "Páginas Web. Características del lenguaje html en el diseño de paginas web. Diseñadores web. Publicidad en Internet," 2008-1995. [Online]. Available: <http://www.hooping.net/faq-caracteristicas.aspx>. [Accessed: 17-Oct-2012].
- [18] A. Froufe, "Tutorial de Java - Arquitectura de appletviewer - Wikilearning," 08-May-2007. [Online]. Available: http://www.wikilearning.com/tutorial/tutorial_de_java-arquitectura_de_appletviewer/3938-37. [Accessed: 17-Oct-2012].
- [19] "phpBB • BBCode guide." .
- [20] R. Bartsch, S. Malaika, and C. Pichler, "Build a DB2 pureXML application in a day," 11-Dec-2008. [Online]. Available: <http://www.ibm.com/developerworks/data/library/techarticle/dm-0812malaika/>. [Accessed: 22-Nov-2012].
- [21] Perz Haide, "Capitulo4," *Diseño e Implementación de un Portal Web*. 2012.
- [22] "The Architecture of Open Source Applications: Python Packaging," *Python Packaging*. [Online]. Available: <http://www.aosabook.org/en/packaging.html>. [Accessed: 25-Oct-2012].
- [23] J. Graham T., A. Ludovic, D. S. Goodsell, M. F. Sanner, and A. J. Olson1, "ScienceDirect.com - Structure - ePMV Embeds Molecular Modeling into Professional Animation Software Environments," 09-Mar-2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0969212611000608>. [Accessed: 25-Oct-2012].
- [24] P. J. D. Harvey M. Deitell, *Cómo programar en C++*, Cuarta. Mexico: Universidades@pearsoned.com, 2003.
- [25] "C++ Preprocessor Directives | CodingUnit Programming Tutorials," 2012. [Online]. Available: <http://www.codingunit.com/cplusplus-tutorial-preprocessor-directives>. [Accessed: 17-Oct-2012].
- [26] H. M. Deitel and P. J. Deitel, *Cómo programar en Java*. Pearson Educación, 2004.
- [27] Froufe Agustín, "Tutorial de Java - Características de Java," 17-May-1999. [Online]. Available: <http://tabasco.torreingenieria.unam.mx/gch/Curso%20de%20Java%20CD/D ocumentos/froufe/parte2/cap2-5.html>. [Accessed: 17-Oct-2012].
- [28] C. de tecnologia informatica (SAO saraoro@alumni.unav.es), "Introducción al lenguaje JAVA," 01-Jul-2000. [Online]. Available: <http://www.unav.es/SI/manuales/Java/indice.html>. [Accessed: 17-Oct-2012].
- [29] "Home | Soka Gakkai International (SGI)," *Soka Gakkai International*, 2012. [Online]. Available: <http://www.sgi.org/index.html>. [Accessed: 16-Oct-2012].

- [30] Snuffer John T., *System and method of extracting 3-D data generated for 2-D display applications for use in 3-D volumetric displays*. United States: , 2004.
- [31] Wernecke Josie, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor (TM) , Release 2*. Keith Wollman, 1994.
- [32] Silicon Graphics, "Chapter 6. Application Libraries," 1998. [Online]. Available: http://techpubs.sgi.com/library/dynaweb_docs/0620/SGL_Developer/books/_IRIX_Prog/sgi_html/ch06.html#id30252. [Accessed: 17-Oct-2012].
- [33] "VTK - The Visualization Toolkit," *VTK Visualization Toolkit*, 2012. [Online]. Available: <http://www.vtk.org/>. [Accessed: 25-Oct-2012].
- [34] "VTK 4.2.1 Documentation," *VTK 4.2.1 Documentation*. [Online]. Available: <http://www.vtk.org/doc/release/4.2/html/>. [Accessed: 25-Oct-2012].
- [35] J. T. Bell, "VTK Tutorial," 2004. [Online]. Available: <http://www.cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html>. [Accessed: 25-Oct-2012].
- [36] R. WANG and X. QIAN, *Openscenegraph 3.0: Beginner's Guide*. Packt Publishing Ltd, 2010.
- [37] L. Montesdeoca Bermúdez and J. Hernández Peter, "DECORACIÓN DE INTERIORES DE OFICINAS USANDO REALIDAD VIRTUAL," 2010.
- [38] G. Hopkins, "introduction," 2011. [Online]. Available: <http://www.java3d.org/introduction.html>. [Accessed: 25-Oct-2012].
- [39] E. PUYBARET, "Du C/C++ à Java : Java 3D," 2007. [Online]. Available: <http://www.eteks.com/coursjava/java3D.html>. [Accessed: 25-Oct-2012].
- [40] "Oracle Technology Network for Java Developers," *Oracle*. [Online]. Available: <http://www.oracle.com/technetwork/java/index.html>. [Accessed: 17-Oct-2012].
- [41] HIPerWorks, "HIPerWorks CGLX Pro Documentation," 2012. [Online]. Available: <http://www.hiperworks.com/pirdoc/cglx-doc/index.html>. [Accessed: 07-Jan-2013].
- [42] "OpenGL - Kcchao," 09-Feb-2010. [Online]. Available: <http://kcchao.wikidot.com/opengl>. [Accessed: 17-Oct-2012].
- [43] Microsoft, "Visual Studio Development Environment Model," 2012. [Online]. Available: <http://msdn.microsoft.com/en-us/library/bb165114.aspx>. [Accessed: 17-Oct-2012].
- [44] G. C. Murphy, M. Kersten, and L. Findlater, "How are Java software developers using the Eclipse IDE?," *IEEE Software*, vol. 23, no. 4, pp. 76 – 83, Aug. 2006.
- [45] Kehn Dan, "Extend Eclipse's Java Development Tools," 22-Jul-2003. [Online]. Available: <http://www.ibm.com/developerworks/java/library/os-ecjdt/>. [Accessed: 17-Oct-2012].

- [46] “Code::Blocks,” 2012. [Online]. Available: <http://www.codeblocks.org/>. [Accessed: 16-Oct-2012].
- [47] Rosales Noé Escobedo, “Manual de 3D Studio Max 8 por el Instituto Tecnológico de Durango,” 15-May-2008. [Online]. Available: <http://www.foro3d.com/f112/manual-3d-studio-max-8-instituto-tecnologico-durango-60725.html?pagenumber=>. [Accessed: 17-Oct-2012].
- [48] “blender.org - Home.” [Online]. Available: <http://www.blender.org/>. [Accessed: 16-Oct-2012].
- [49] R. H. Shih, *AutoCAD 2013 Tutorial: First Level: 2D Fundamentals*. SDC Publications, 2012.
- [50] “Chapter 1,” *Introduction to AutoCAD*. [Online]. Available: <http://www.ktscss.edu.hk/subject/technology/designandtechnology/a05/command-acad/ac01.htm>. [Accessed: 17-Oct-2012].
- [51] “Descargar Photoshop gratis,” 11-Jun-2011. [Online]. Available: <http://photoshop.soft2k.com/>. [Accessed: 17-Oct-2012].
- [52] M. N. D. Bernal C. Iván and Fernández A. Diego, “Computación de Alto Rendimiento con Clusters de PCs.” .
- [53] “OpenClusterGroup.org,” 2006-2000. [Online]. Available: <http://www.openclustergroup.org/>. [Accessed: 17-Oct-2012].
- [54] J. D. Sloan, *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI: With OSCAR, Rocks, openMosix, and MPI*, First. O’Reilly Media, Inc., 2004.
- [55] “StackIQ Home Page,” 2012. [Online]. Available: <http://www.stackiq.com/>. [Accessed: 17-Oct-2012].
- [56] A. Guerrero, “Página de inicio | Coordinación General de Tecnologías de Información.” [Online]. Available: <http://www.cgti.udg.mx/>. [Accessed: 25-Oct-2012].
- [57] “openMosix,” *SourceForge*, 2010. [Online]. Available: <http://sourceforge.net/projects/openmosix/>. [Accessed: 25-Oct-2012].
- [58] “Sistema de computación masiva en Sun Grid.” [Online]. Available: http://rdlab.lsi.upc.edu/docu/html/cluster/#_Toc219597779. [Accessed: 25-Oct-2012].
- [59] “que es un applet?” [Online]. Available: <http://ict.udlap.mx/people/roberto/cursojava/applet.html>. [Accessed: 25-Oct-2012].
- [60] “Объективные недостатки C++,” 12-Jan-2011. [Online]. Available: <http://habrahabr.ru/post/111708/>. [Accessed: 25-Oct-2012].
- [61] “Vor- und Nachteile von JAVA,” 2010. [Online]. Available: http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/SeminarDidaktik/OOP/java_vor_nachteile.html. [Accessed: 25-Oct-2012].
- [62] Molina Carmona Rafael and Puchol García Juan Antonio, “APUNTES DE OPENGL.” 99-1998.

- [63] “MIT OpenCourseWare | Civil and Environmental Engineering | 1.124J Foundations of Software Engineering, Fall 2000 | Lecture Notes | Java 3D Lecture,” 2012-2002. [Online]. Available: http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-124j-foundations-of-software-engineering-fall-2000/lecture-notes/java_3d_lecture/#1. [Accessed: 25-Oct-2012].
- [64] F. Hanisch, “European Journal of Open, Distance and E-Learning,” 2000. [Online]. Available: <http://www.eurodl.org/?p=archives&year=2000&article=94>. [Accessed: 25-Oct-2012].
- [65] L. J. Víctor and P. M. José, “Java 3D y Looking Glass Desarrollos open source de SUN,” Escuela Politécnica Superior de Albacete, 12 diciembre-2004.
- [66] “Creación de ambientes virtuales inmersivos con software libre - María del C. Ramos, José Larios, Daniel Cervantes y Renato Leriche,” 2012. [Online]. Available: <http://www.oei.es/noticias/spip.php?article823>. [Accessed: 25-Oct-2012].
- [67] “Eclipse - The Eclipse Foundation open source community website.,” 2012. [Online]. Available: <http://www.eclipse.org/>. [Accessed: 16-Oct-2012].
- [68] M. Khouzam, “CDT/User/NewIn70 - Eclipsepedia,” 2012. [Online]. Available: <http://wiki.eclipse.org/CDT/User/NewIn70>. [Accessed: 25-Oct-2012].
- [69] E. Vértice, *Photoshop*. Editorial Vértice, 2010.
- [70] D. Giménez, M. Boratto, and L. Coelho, “Programación Paralela y Distribuida.” 2009.
- [71] B. Blaise, “Message Passing Interface (MPI),” 07-Oct-2012. [Online]. Available: <https://computing.llnl.gov/tutorials/mpi/>. [Accessed: 25-Oct-2012].
- [72] L. Fialho, “Incorporando RADICa OpenMPI.” .
- [73] F. H. NETTER, *Atlas de Anatomía Humana*, 4th ed. Barcelona (España): Elsevier Masson, 2007.
- [74] Tripod, “Modelado Gráfico - Java 3D.” 2012.
- [75] “www.rocksclusters.org | Rocks Website,” 2012. [Online]. Available: <http://www.rocksclusters.org/wordpress/>. [Accessed: 17-Oct-2012].
- [76] Ángel Acaymo M. G., “Servidor Web Apache, PHP, MySQL.,” 06-Jun-2009.
- [77] GRAVITY, “CGLX Project.” 2008.
- [78] “Quick Start.” [Online]. Available: <http://www.hiperworks.com/pirdoc/qstart-doc/index.html>. [Accessed: 22-Nov-2012].
- [79] T. Boardman, *3Ds Max 6 Fundamentals*. New Riders, 2004.
- [80] oracle, “Java Archive Downloads -Java Client Technologies,” 2013. [Online]. Available: <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java->

archive-downloads-java-client-419417.html#java3d-1.4.0_01-oth-JPR.
[Accessed: 26-Jan-2013].

- [81] "Eclipse Downloads," 2013. [Online]. Available: <http://www.eclipse.org/downloads/>. [Accessed: 26-Jan-2013].
- [82] "Introducción a Java3D (página 3) - Monografias.com," 2012. [Online]. Available: <http://www.monografias.com/trabajos43/java-tres-d/java-tres-d3.shtml>. [Accessed: 07-Feb-2013].
- [83] "Downloading File libserial-0.5.0.tar.gz - libserial - SourceForge.JP," *Serial Port*, 2012. [Online]. Available: http://en.sourceforge.jp/projects/sfnet_libserial/downloads/libserial/libserial-0.5.0/libserial-0.5.0.tar.gz/. [Accessed: 02-Feb-2013].

5. ANEXOS

5.1 ANEXO A – HU1: CREACIÓN Y EDICIÓN DE IMÁGENES EN 3D

5.1.1 Instalación de 3D Max Studio 7

Pasos a seguir:

Para abrir una aplicación de consola dirigirse a: Aplicaciones/Accesorios/Terminal.

Para restaurar la lista de paquetes se procede a realizar una actualización con el siguiente comando en Ubuntu 11.04:

```
$ sudo apt-get update
```

Luego se procederá a instalar el paquete *Wine* ya que este es el programa que permitirá la ejecución desde Ubuntu 11.04.

```
$ sudo apt-get install Wine
```

Se mostrará la siguiente pantalla:

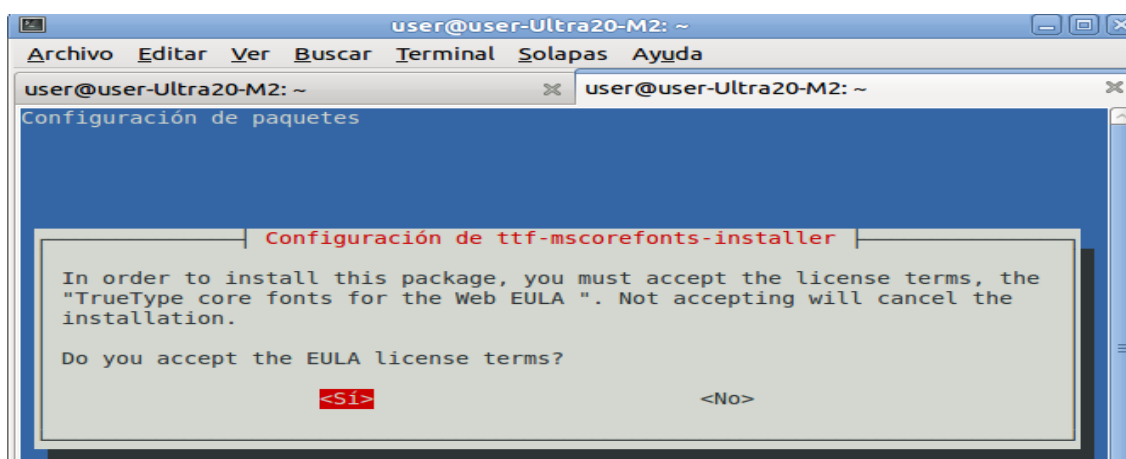


Figura A - 1 Configuración del paquete "ttf"

Fuente: Autores

Para configurar dirigirse a *Aplicaciones / Wine / Configure Wine* y tipear:

```
$ winecfg
```

Se abrirá la siguiente pantalla donde se colocará aceptar:

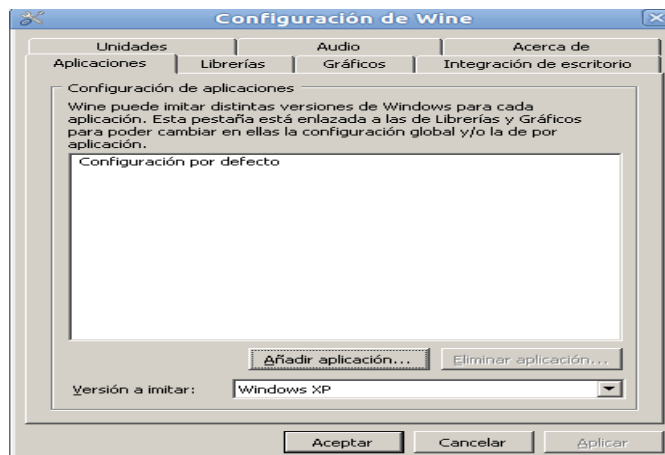


Figura A - 2 Configuración del WINE

Fuente: Autores

En el directorio del programa, tipear lo siguiente para obtener la última versión e instalar un complemento de los paquetes Microsoft:

```
$ cd .wine/
$ sudo wget http://www.kegel.com/wine/winetricks
$ sudo apt-get install cabextract
```

Para instalar el .net framework tipear la siguiente línea:

```
$ sh winetricks dotnet20 allfonts
```

Aceptar la licencia e instalar, una vez finalizada se podrá ejecutar el 3D Max 7 pero sobre Ubuntu 11.04

Para instalar un ejecutable que esté con extensión .exe se lo ejecutará con “*Wine core exe*”, “*winhlp32*” o “*Wine application*”.


Dar clic en el icono de 3D Max Studio 7  y seleccionar “*winhlp32*”:

Aceptar las condiciones de la licencia en la ventana respectiva y dar clic en “*Next*”. Aparece una nueva ventana que pide se introduzca el serial, no introducirlo solo dar *Next* y continuar:

Figura A - 3 Información de usuario de 3D max7 y número serial

Fuente: Autores

Esperar y una vez finalizada la instalación aparecerá en escritorio el logo de

3DMax 7  con lo cual ya se podrá trabajar.

Terminada la instalación dirigirse a:

Sistema de Archivos y en la barra de herramientas en Ver seleccionar Mostrar los archivos ocultos buscar el archivo *stdplug* el cual se encuentra en la siguiente ruta:

```
home / usuario / .wine / dosdevices / c: / 3dsmax7 / stdplugins
```

En la ubicación indicada buscar y borrar el archivo *cmbtex.dlt*, de lo contrario 3D Max 7 no arrancará, se quedará cargando en memoria pero nunca iniciará.

Verificación:

Hecho esto se ejecutará el programa dando doble clic:



Figura A - 4 Ventana de ingreso a 3D max7

Fuente: Autores

Detrás de esta ventana aparece la primera pantalla de la Figura A - 5 la cual se la va a mover con el mouse y seleccionar *Run the product* porque el programa aún no está activado, luego dar clic en *next* y en la segunda pantalla que aparece, seleccionar la opción *Software*, si luego aparece otra ventana reportando algún error presionar aceptar:

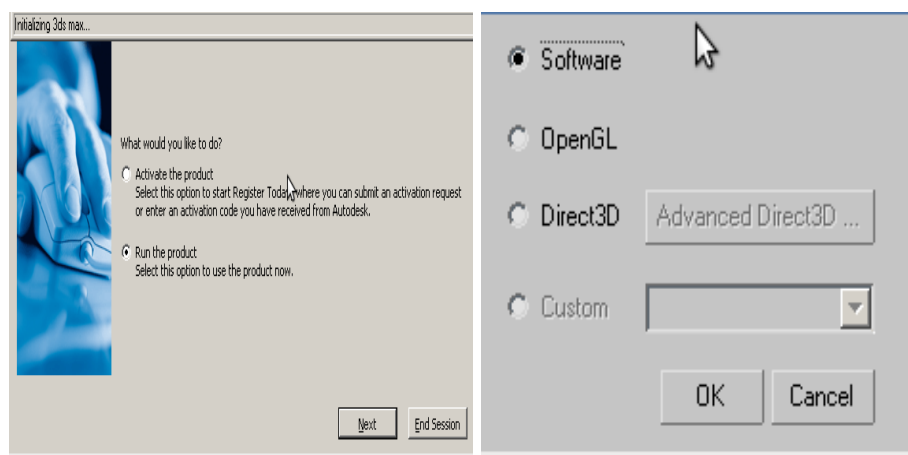


Figura A - 5 Ventana de inicialización para ejecutar o activar el producto

Fuente: Autores

Listo para su funcionamiento se arrastrará o importará una imagen y se la podrá visualizar:

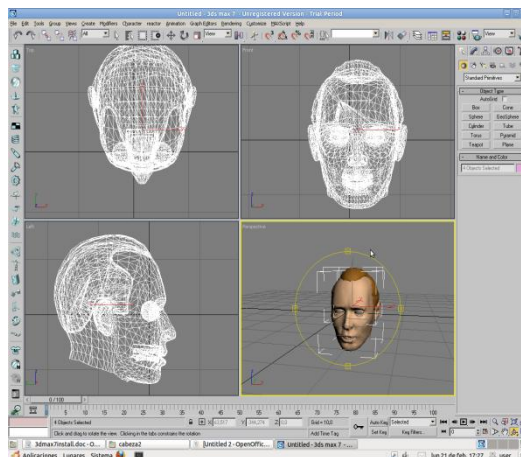


Figura A - 6 Ejecución correcta del funcionamiento de 3D max7

Fuente: Autores

5.1.2 Herramientas de 3D Max Studio 7

A continuación se describen algunas de las herramientas a utilizar para el diseño de las partes del cuerpo humano que se consideraron prudentes tomadas de libros y documentos.[79]

A. Barras de menús.

(File, Edit, Tool, Group, Views) muestran los menús que se utilizan en 3D, en donde se desplegarán otras opciones

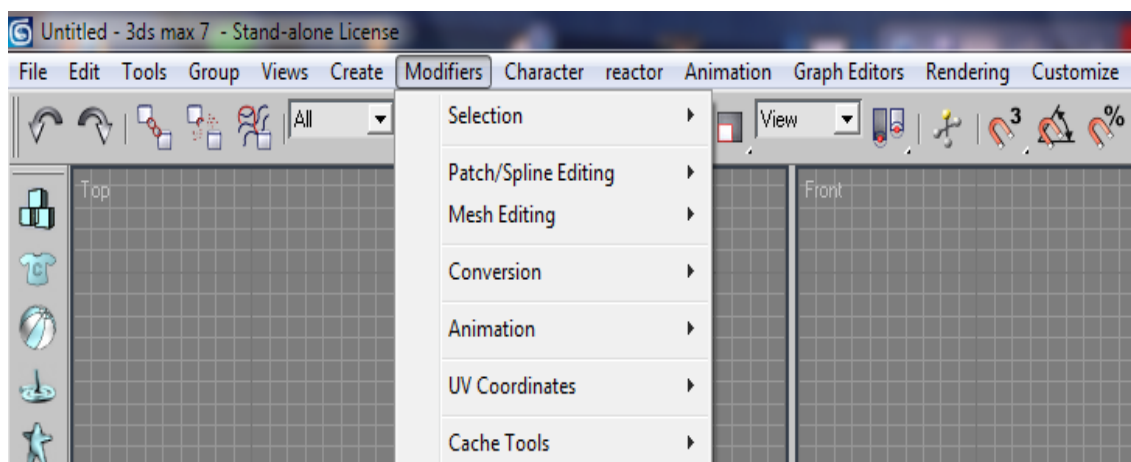


Figura A - 7 Barras de Menú Autodesk 3D Max

Fuente: Autores

B. Barra de herramientas.

Permite el acceso rápido a herramientas y cuadros de diálogo para muchas tareas comunes



Figura A - 8 Barras de Herramientas Autodesk 3D Max

Fuente: Autores

C. Transform Toolbars (Barra de transformaciones)



MOVE (Mover).- Permite desplazar los objetos, si se presiona F12, se despliega el conmutador de transformaciones en el cual se puede especificar las coordenadas donde se quiere que el objeto se sitúe.



ROTATE (Rotar).- Permite hacer rotaciones, en los distintos ejes de simetría



ESCALE (Escalar).- Permite reducir o aumentar el tamaño del objeto mediante un porcentaje de escala, hay 2 formas de escalar objetos: uniforme el objeto conserva la proporción, no uniforme, el objeto no conserva la proporción, puede ser escalado en los 3 ejes de simetría independientemente.

Para visualizar el resto de botones colocar el cursor en la orilla de la barra y al aparecer una mano, moverla hacia la izquierda o derecha con lo que se desplegarán nuevas herramientas como: material editor, renderscene, etc.



Figura A - 9 Barras de Herramientas 2 Autodesk 3D Max

Fuente: Autores

D. Barra de editores.Layer Manager (Administrador de capas)



Curve Editor (Editor de curvas). Para visualizar y modificar los parámetros de un objeto a través del tiempo, es decir, todo lo que pueda ser animado (cambiado con el tiempo).



Schematic View (Editor de vista esquemática).- Es una ventana que muestra en forma de esquemas los subconjuntos de los objetos de la escena en un gráfico.



Material Editor (Editor de materiales). Opción que proporciona funciones para crear y editar materiales y mapas. Un material es un dato que se asigna a la superficie o las caras de un objeto para que aparezca de una determinada manera cuando se muestra. El material afecta al color del objeto en brillo, opacidad, etc. Las imágenes que se asignan a los materiales se llaman mapas.

Panel de solapas. Proporciona comandos, controles y parámetros para crear, modificar, vincular, animar y presentar diseños de objetos.



Figura A - 10 Panel de solapas Autodesk 3D Max

Fuente: Autores

E. Herramientas del panel de solapas:



Create. Permite crear casi todos los elementos entre los cuales se encuentran:



Geometry. Geometría. Da acceso a las primitivas de 3D Studio Max en 3D, que pueden ser: esferas, cajas, cilindros.



Shapes. Formas planas. Da acceso a las primitivas planas del programa, son líneas poligonales, rectángulos, arcos, etc.



Lights. Luces. Comando para crear las luces con las que se iluminará la escena. En el momento que creamos una luz, la luz por defecto del programa desaparecerá. Estas luces pueden ser: omnidireccionales, focos, etc.



Cameras. Cámaras. Sirve para crear las cámaras de la escena. Una vez que se haya creado una cámara, seleccionar una viewport y pulsar la tecla C y la viewport se transformará en la vista que se ve en la cámara.



Helpers. Ayudantes. Da acceso a los ayudantes: cinta métrica, rejilla, etc.



Space warps. Espacios de modificación. Este botón da acceso a los comandos de efectos especiales como: bomba, honda, viento, etc.



Systems. Sistemas. Esto sirve para los huesos de animación, matriz tridimensional, etc.



Modify. Es de los más importantes porque con este se va a modificar todo lo que se puede crear en el programa, ya que podemos cambiar el aspecto de un objeto así como los parámetros referentes a materiales y cámaras. Al dar clic en este botón cambia la solapa de abajo y aparecen botones los cuales se llaman modificadores.



Hierarchy. Muestra las opciones cuando los objetos están vinculados entre sí, como también las diferentes opciones del pivote del objeto el cual se puede ajustar a conveniencia.



Motion. Contiene información de movimientos de los objetos animados como su trayectoria desde el punto inicial al punto final.



Display. Permite definir las características del objeto mostrado en los visores como: ocultar, congelar, mostrar propiedades del objeto, etc.



Utilities. Contiene diversas opciones principales de plugins como: reactor, maxscrip.

F. Barra de exploración de visores.

Proporciona una manera de movimiento o de percepción de objetos dentro de las vistas en las que se trabajará permitiendo explorar la escena, mediante:



Zoom. Acerca o aleja la visualización del objeto.



Zoom All. Zoom Todo. Permite observar todos los objetos dentro de los distintos visores.



Zoom Extended. Zoom a extensión. Realiza un zoom, de forma que los objetos contenidos en los visores sean visibles abarcando los distintos visores.

Zoom Extended Objet. Zoom a extensión seleccionado. Realiza un zoom solamente al objeto seleccionado de manera que este abarque los visores.



Field Of View. Campo visual. Amplian o disminuyen el campo visual.



Pan. Encuadrar visor. Permite arrastrar la orientación de los objetos para ubicarlos donde se requiera.



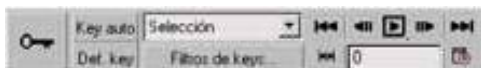
Rotate. Rotar. Opción que permite rotar las distintas vista contenidas en los diferentes visores, principalmente en la vista perspectiva.



Zoom Window. Zoom por ventana. Permite seleccionar un rectángulo, en el cual se centrará el acercamiento.



Viewport Toggle Max/Min. Conmutador min. / max. Permite cambiar entre visualizar los cuatro visores a visualizar un visor en específico.



Control de tiempo. Permite controlar los movimientos, animar una película y moverse por los fotogramas que constituyen las mismas.

5.1.3 Añadir Texturas a objetos 3D

- Importar el elemento 3D con extensión .obj sin librerías (material) a 3D Max Studio
- Seleccionar todo el elemento .obj con la herramienta “Select Object”, ir a la pestaña de modificaciones “Modifiers List”, seleccionar y añadir “Unwrap UVW” (ver Figura 3-16)
- En la opción de “Parameters” seleccionar “Edit” posteriormente se abrirá una nueva ventana
- Seleccionar al objeto completamente
- En la pestaña “Select” elegir: “Select Vertex to Face” y después en la pestaña “Mapping” elegir “mapping normal...” y en la ventana que aparece poner “OK”

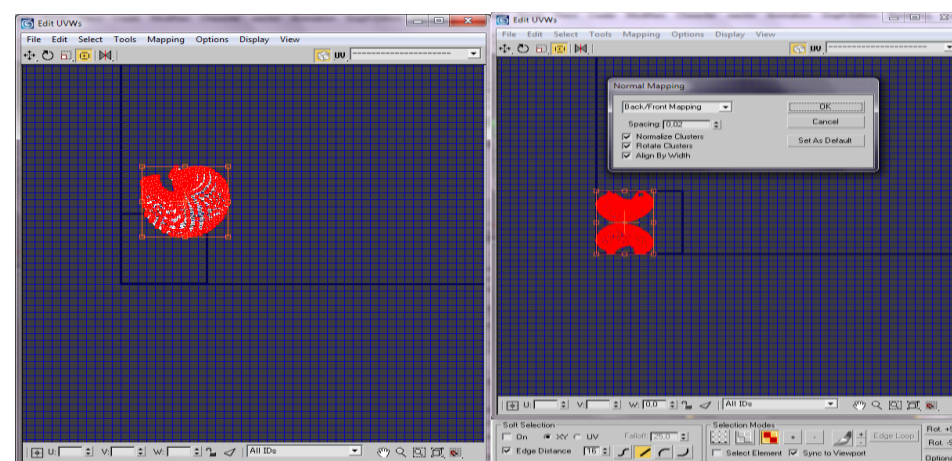


Figura A - 11 Ventana de Mapping

Fuente: Autores

- Ir a “Tools” y seleccionar “Render UVW Template...” se abrirá una ventana y dar clic en el botón inferior “Render UV Template”. Guardar deseleccionando la opción de “Alpha channel”
- En Adobe Photoshop abrir la imagen .png que se exportará desde 3D_Max Studio

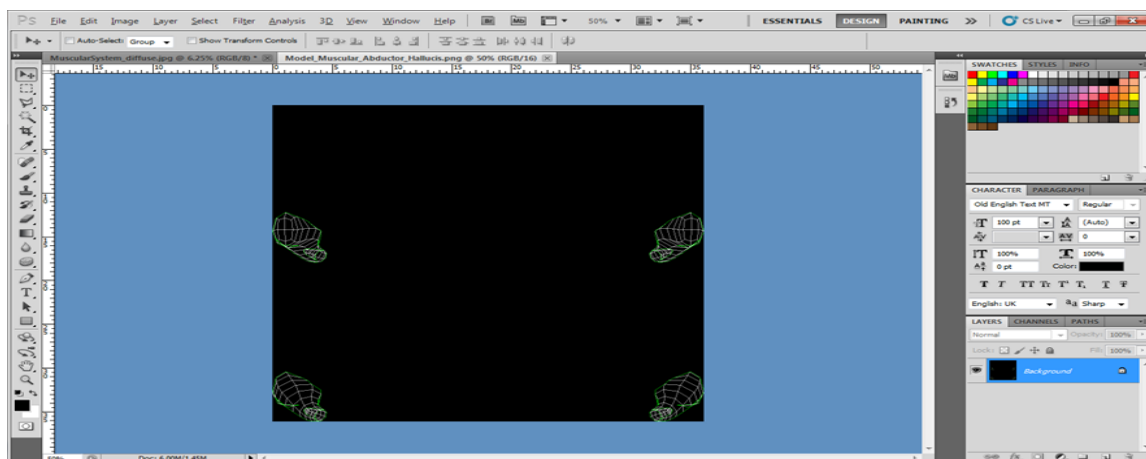


Figura A - 12 Ventana de edición de la imagen en Adobe Photoshop

Fuente: Autores

- Editar la imagen añadiendo las texturas y guardar el archivo con extensión .jpg

5.1.4 Fusión Diseño 3D-Texturas

Dar clic en el primer casillero y en la parte inferior dar clic frente a “Diffuse” con lo que se despliega otra ventana en la cual se seleccionará la ubicación donde se encuentra el archivo JPG a utilizar como material de relleno del objeto 3D

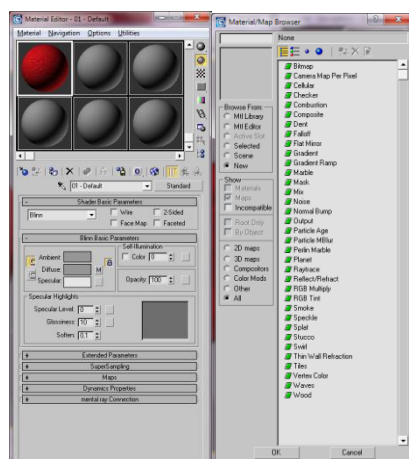


Figura A - 13 Ventana de selección del archivo a abrir.

Fuente: Autores

Doble clic en “Bitmap” con lo cual aparece la ventana de selección para elegir la imagen a colocar como textura y abrir

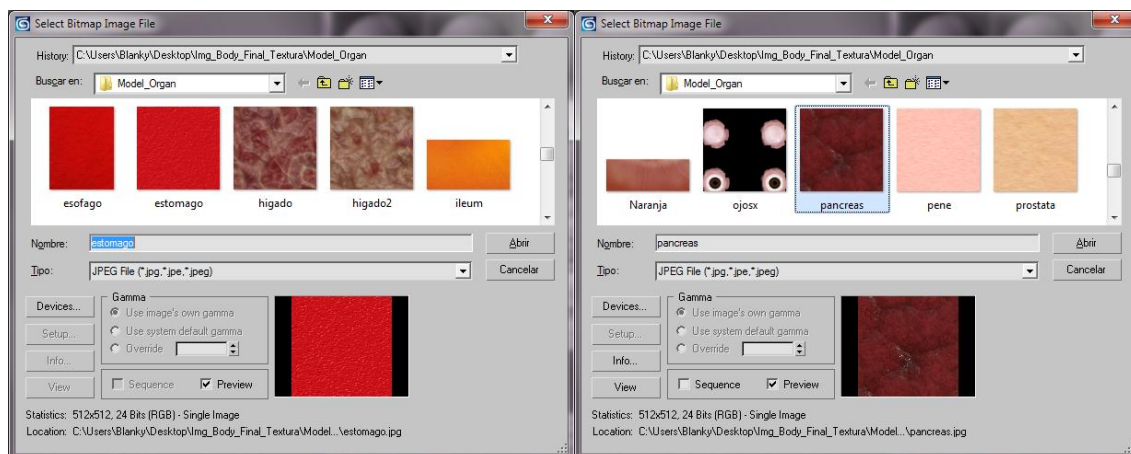


Figura A - 14 Ventana para elegir la textura

Fuente: Autores

Se visualiza que el material ha cubierto el primer recuadrado del editor de materiales, para luego ser aplicado a la imagen.



Dar clic en “Asignar Material a Selección” y “Show Map in Viewport” para colocar la textura en la imagen .obj, con esto se consigue colocar materiales a distintos objetos 3D ver Figura 3-17

5.2 ANEXO B – HU2: CREACIÓN DE UN APLICATIVO PARA MANIPULAR IMÁGENES 3D

5.2.1 Aplicativo con Java 3D

5.2.1.1 Configuración de JRE

Descargar el archivo Java JRE desde Internet[40]

Si se tiene una versión anterior de **JRE** configurada en el sistema de forma manual se la borrará para evitar una duplicidad de versiones.

```
$ su
# rm -rf /usr/java/{jre1.7.0_02,default}
# rmdir --ignore-fail-on-non-empty /usr/java &> /dev/null
```

Se procederá a extraer el paquete en el lugar establecido y se creará el enlace simbólico por defecto (default), que se utilizará para establecer la variable de entorno PATH de Java.

```
$ su
# mkdir -p /usr/java
# tar zxvf jre-7u3-linux-i586.tar.gz -C /usr/java
# cd /usr/java
# ln -s jre1.7.0_03 default
# cp -rf default/lib/desktop/* /usr/share
# update-mime-database /usr/share/mime
# for i in HighContrast HighContrastInverse LowContrast hicolor ; do \
gtk-update-icon-cache -t /usr/share/icons/$i ; \
done
```

```

root@migi: /usr/java
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@migi:/home/migi/Descargas# cd /usr/java
root@migi:/usr/java# ln -s jre1.7.0_03 default
root@migi:/usr/java# cp -rf default/lib/desktop/* /usr/share
root@migi:/usr/java# update-mime-database /usr/share/mime
Unknown media type in type 'all/all'
Unknown media type in type 'all/allfiles'
Unknown media type in type 'uri/mms'
Unknown media type in type 'uri/mmst'
Unknown media type in type 'uri/mmsu'
Unknown media type in type 'uri/pnm'
Unknown media type in type 'uri/rtsp'
Unknown media type in type 'uri/rtspu'
Unknown media type in type 'interface/x-winamp-skin'
root@migi:/usr/java# for i in HighContrast HighContrastInverse LowContrast hicolor ; do \
> gtk-update-icon-cache -t /usr/share/icons/$i ; \
> done
gtk-update-icon-cache: Cache file created successfully.
gtk-update-icon-cache: Cache file created successfully.
gtk-update-icon-cache: Cache file created successfully.
gtk-update-icon-cache: Cache file created successfully.
root@migi:/usr/java#

```

Figura B - 1 Configuración del JRE

Fuente: Autores

Añadir la dirección (path) JRE

Se hará uso del enlace simbólico “/usr/java/default” para no editar este archivo cada vez que se actualice el entorno Java.

Editar el archivo de configuración personal de “Bash, ~/.bashrc”. Si no existe; crearlo e incluir el “PATH” para facilitar la ejecución de los binarios al final del mismo. Para ello en un terminal colocar: “gedit ~/.bashrc” o “nano ~/.bashrc” con el que se permitirá crear el archivo.

```

export PATH=/usr/java/default/bin:$PATH
export JAVA_HOME=/usr/java/default

```

Para comprobar que el binario Java está incluido en el “Path”, abrir una ventana de terminal y ejecutar el siguiente comando:

```
java -version
```

Para establecer una variable de entorno global del sistema, abrir un editor de texto y añadir lo siguiente:

```

#!/bin/sh
export PATH=/usr/java/default/bin:$PATH
export JAVA_HOME=/usr/java/default

```

Guardar con el nombre “java.sh”, e instalar en “/etc/profile.d”.

```
$ su
# install -m755 java.sh /etc/profile.d
```

Cerrar el terminal y volverlo a abrir para que de esta manera la variable de entorno se aplique.

5.2.1.2 Configuración de JDK

Primero se debe descargar el archivo Java JDK desde Internet [40].

Si se tiene una versión anterior de **JDK** configurada en el sistema de forma manual borrar para evitar duplicidad de versiones

```
$ su
# rm -rf /usr/java/{jdk1.7.0_02,default}
# rmdir --ignore-fail-on-non-empty /usr/java &> /dev/null
```

Se procederá a extraer el paquete en el lugar indicado y luego se creará el enlace simbólico “default”, que se utilizará para establecer la variable de entorno PATH de Java:

```
$ su
# mkdir -p /usr/java
# tar zxvf jdk-7u3-linux-i586.tar.gz -C /usr/java
# cd /usr/java
# ln -s jdk1.7.0_03 default
# cp -rf default/jre/lib/desktop/* /usr/share
# update-mime-database /usr/share/mime
# for i in HighContrast HighContrastInverse LowContrast hicolor ; do \
gtk-update-icon-cache -t /usr/share/icons/$i ; \
done
```

La Figura B - 2 Configuración de JDK muestra lo indicado:

```

jdk1.7.0_03/jre/bin/rmid
jdk1.7.0_03/jre/COPYRIGHT
jdk1.7.0_03/jre/Welcome.html
jdk1.7.0_03/release
jdk1.7.0_03/COPYRIGHT
root@migi:/home/migi/Descargas# cd /usr/java
root@migi:/usr/java# ln -s jdk1.7.0_03 default
root@migi:/usr/java# cp -rf default/jre/lib/desktop/* /usr/share
cp: no se puede efectuar `stat' sobre «default/jre/lib/desktop/*»: No existe el
fichero o el directorio
root@migi:/usr/java# update-mime-database /usr/share/mime
Unknown media type in type 'all/all'
Unknown media type in type 'all/allfiles'
Unknown media type in type 'uri/mms'
Unknown media type in type 'uri/mmst'
Unknown media type in type 'uri/mmsu'
Unknown media type in type 'uri/pnm'
Unknown media type in type 'uri/rtsp'
Unknown media type in type 'uri/rtspu'
Unknown media type in type 'interface/x-winamp-skin'
root@migi:/usr/java# for i in HighContrast HighContrastInverse LowContrast hicol
or ; do \
> gtk-update-icon-cache -t /usr/share/icons/$i ; \
> done
root@migi:/usr/java#

```

Figura B - 2 Configuración de JDK

Fuente: Autores

Añadir la dirección (path) del JDK

Se hará uso del enlace simbólico “/usr/java/default” para no editar este archivo cada vez que se actualice el entorno Java.

Editar el archivo de configuración personal de “Bash, ~/.bashrc”. Si no existe; crearlo e incluir el “PATH” para facilitar la ejecución de los binarios al final del mismo. Para ello en un terminal colocar: “gedit ~/.bashrc” o “nano ~/.bashrc”:

```

export PATH=/usr/java/default/bin:/usr/java/default/db/bin:$PATH
export JAVA_HOME=/usr/java/default
export DERBY_HOME=/usr/java/default/db

```

Para comprobar que el binario “javac” está incluido en el “Path”, abrir una ventana en el terminal y ejecutar el siguiente comando:

```
$ javac -version
```

Para establecer una variable de entorno global del sistema, abrir un editor de texto y añadir lo siguiente:

```
#!/bin/sh
export PATH=/usr/java/default/bin:/usr/java/default/db/bin:$PATH
export JAVA_HOME=/usr/java/default
export DERBY_HOME=/usr/java/default/db
```

Guardar con el nombre “java.sh”, e instalar en “/etc/profile.d”:

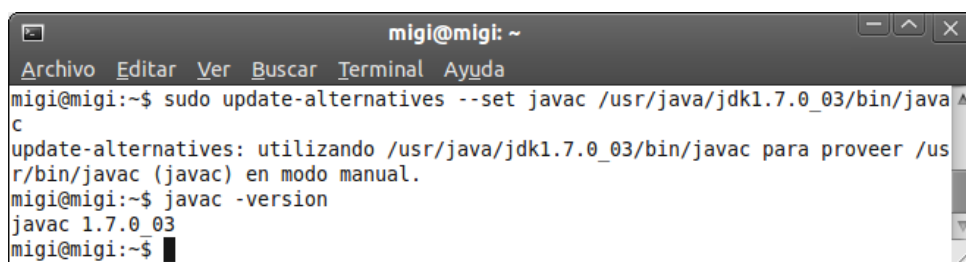
```
$ su
# install -m755 java.sh /etc/profile.d
```

Cerrar el terminal y volverlo a abrir para que la variable de entorno aplicada sea efectiva. En caso de que no se reconozca la versión instalada de “javac” utilizar esta sentencia:

```
$ sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/java/jdk1.7.0_03/bin/javac" 50
```

```
$ sudo update-alternatives --set javac "/usr/java/jdk1.7.0_03/bin/javac"
```

Para comprobar que la versión está actualizada escribir: javac -version



```
migi@migi: ~
Archivo Editar Ver Buscar Terminal Ayuda
migi@migi:~$ sudo update-alternatives --set javac /usr/java/jdk1.7.0_03/bin/javac
update-alternatives: utilizando /usr/java/jdk1.7.0_03/bin/javac para proveer /usr/bin/javac (javac) en modo manual.
migi@migi:~$ javac -version
javac 1.7.0_03
migi@migi:~$
```

Figura B - 3 Actualización del Javac

Fuente: Autores

5.2.1.3 Configuración Java 3D - API 1.4.0_01

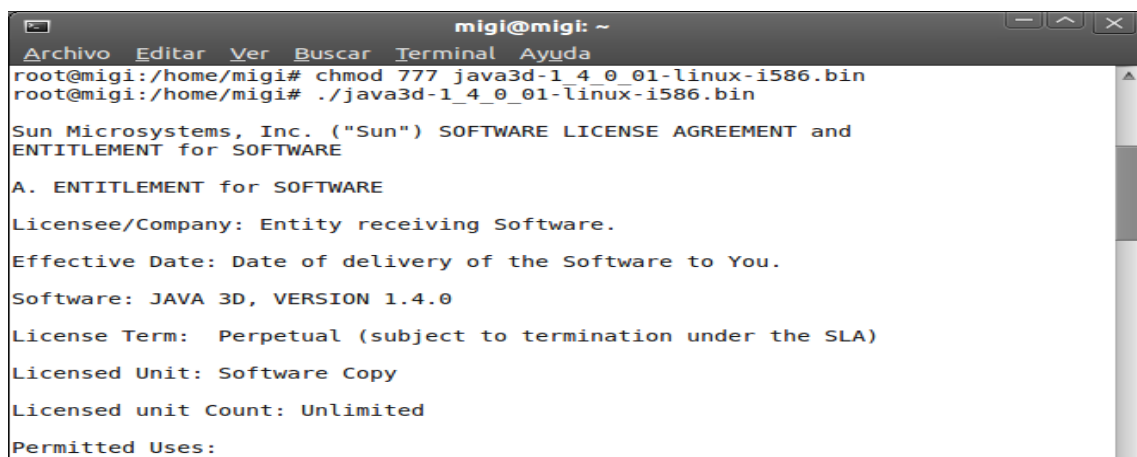
Descargar el archivo de Internet [80]

Una vez descargado el API colocarlo en una ubicación accesible para proceder a dar permisos de lectura y escritura al archivo, de la siguiente manera:

```
# chmod 777 java3d-1_4_0_01-linux-i586.bin
```

Se ejecuta el archivo binario colocando el siguiente comando (ver Figura B - 4 y Figura B - 5):

```
# ./java3d-1_4_0_01-linux-i586.bin
```



```
migi@migi: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@migi:/home/migi# chmod 777 java3d-1_4_0_01-linux-i586.bin
root@migi:/home/migi# ./java3d-1_4_0_01-linux-i586.bin

Sun Microsystems, Inc. ("Sun") SOFTWARE LICENSE AGREEMENT and
ENTITLEMENT for SOFTWARE

A. ENTITLEMENT for SOFTWARE

Licensee/Company: Entity receiving Software.

Effective Date: Date of delivery of the Software to You.

Software: JAVA 3D, VERSION 1.4.0

License Term: Perpetual (subject to termination under the SLA)

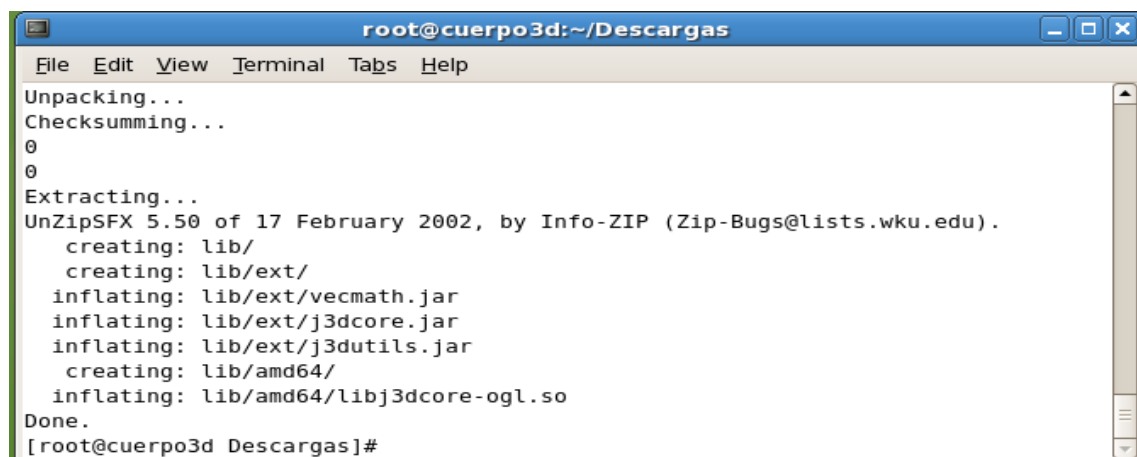
Licensed Unit: Software Copy

Licensed unit Count: Unlimited

Permitted Uses:
```

Figura B - 4 Ejecución del archivo binario en Ubuntu 11.04

Fuente: Autores



```
root@cuerpo3d:~/Descargas
File Edit View Terminal Tabs Help
Unpacking...
Checksumming...
0
0
Extracting...
UnZipSFX 5.50 of 17 February 2002, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  creating: lib/
  creating: lib/ext/
inflating: lib/ext/vecmath.jar
inflating: lib/ext/j3dcore.jar
inflating: lib/ext/j3dutils.jar
  creating: lib/amd64/
inflating: lib/amd64/libj3dcore-ogl.so
Done.
[root@cuerpo3d Descargas]#
```

Figura B - 5 Ejecución del archivo binario en CentOS

Fuente: Autores

Para que se ejecute el Applet 3D en Eclipse, las librerías 'libj3dcore-ogl.so' y 'libj3dcore-ogl-cg.so' extraídos del 'java3d-1_4_0_01' deberán ser copiadas dentro del enlace simbólico "default" que se creó al configurar el JRE y JDK, de esta manera no generará error al no encontrar la ruta de la librería

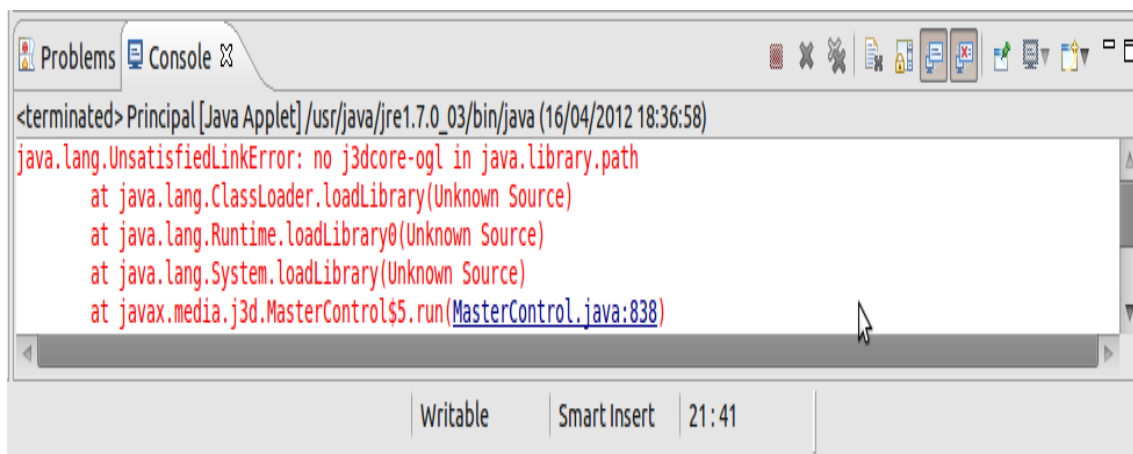


Figura B - 6 Error de no encontrar la ruta de la librería 'libj3dcore-ogl.so'

Fuente: Autores

Abrir un Terminal y copiar de uno en uno los archivos a la ubicación especificada (ver Figura B - 7) con los siguientes comandos:

```
#cp -r /home/migi/lib/ext/j3dcore.jar /usr/java/default/lib/ext
# cp -r /home/migi/lib/ext/j3dutils.jar /usr/java/default/lib/ext
# cp -r /home/migi/lib/ext/vecmath.jar /usr/java/default/lib/ext
# cp -r /home/migi/lib/i386/libj3dcore-ogl.so /usr/java/default/lib/i386
# cp -r /home/migi/lib/i386/libj3dcore-ogl-cg.so /usr/java/default/lib/i386
# cp -r /home/migi/lib/i386/libj3dutils.so /usr/java/default/lib/i386
```

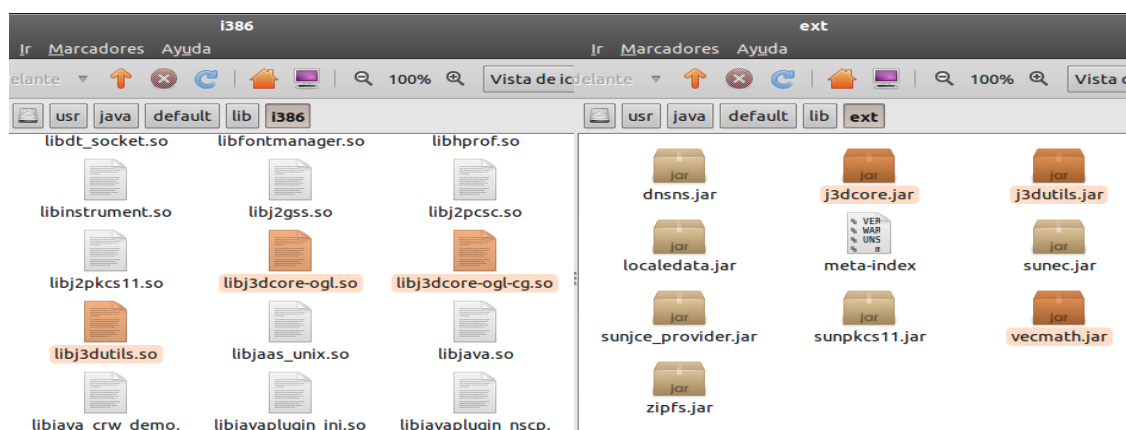


Figura B - 7 Archivos ubicados en la ruta específica dentro de 'i386' y dentro del ext

Fuente: Autores

Para que el navegador Mozilla Firefox, reconozca a Java (J3D) y puedan ser ejecutados los Applets 3D, se procede a enlazar de manera simbólica la

carpeta “*.mozilla/plugins*”, siendo necesario que el navegador de Mozilla este previamente cerrado.

En caso de tener enlaces previos a Mozilla, borrarlos de la siguiente manera:

```
rm -rf .mozilla/plugins/libnjp2.so
rm -rf .mozilla/plugins/libjavaplugin_oji.so
```

Si no se cuenta con una carpeta para el plugin se creará con la siguiente instrucción:

```
$ sudo mkdir .mozilla/plugins
```

Enlazar a Mozilla las librerías: *libjavaplugin_oji.so* y *libnjp2.so* creando los vínculos al plugin de Java como indica la Figura B - 8

```
$ cd .mozilla/plugins/
$ sudo ln -s /usr/java/jre1.7.0_03/lib/i386/libnjp2.so
$ sudo ln -s /usr/java/jre1.7.0_03/plugin/i386/ns7/libjavaplugin_oji.so
```

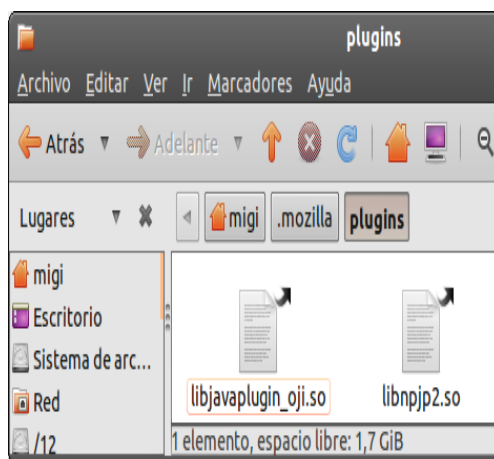


Figura B - 8 Archivos ubicados en la ruta especificada en Mozilla.

Fuente: Autores

5.2.1.4 Instalación de Eclipse para Ubuntu

La instalación de Eclipse se realiza desde el “Centro de Software” de Ubuntu

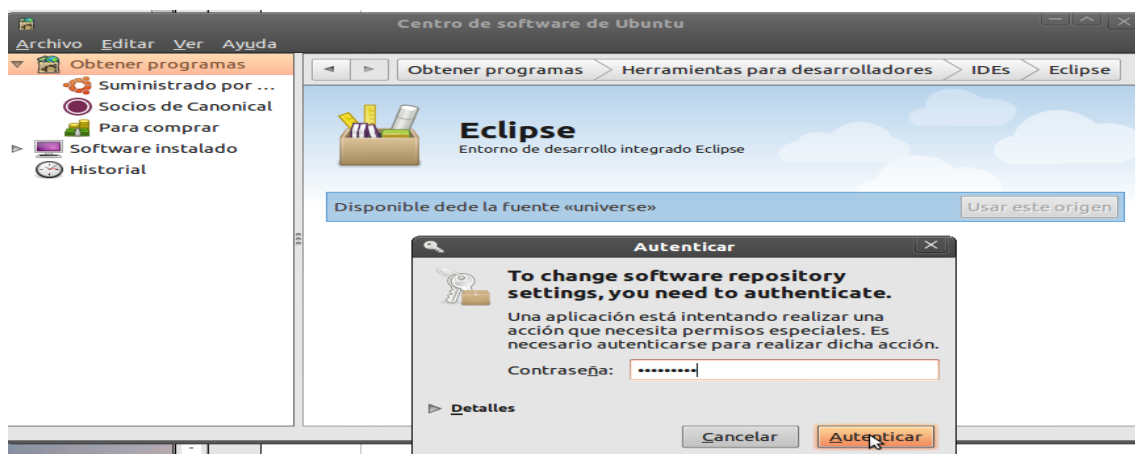


Figura B - 9 Instalación de Eclipse

Fuente: Autores

Instalar “Java 3DS File Loader” dirigiéndose a “Sistema/ Gestor de Paquetes Synaptic”, en la ventana que aparece escribir el nombre del archivo y dar clic en aplicar:

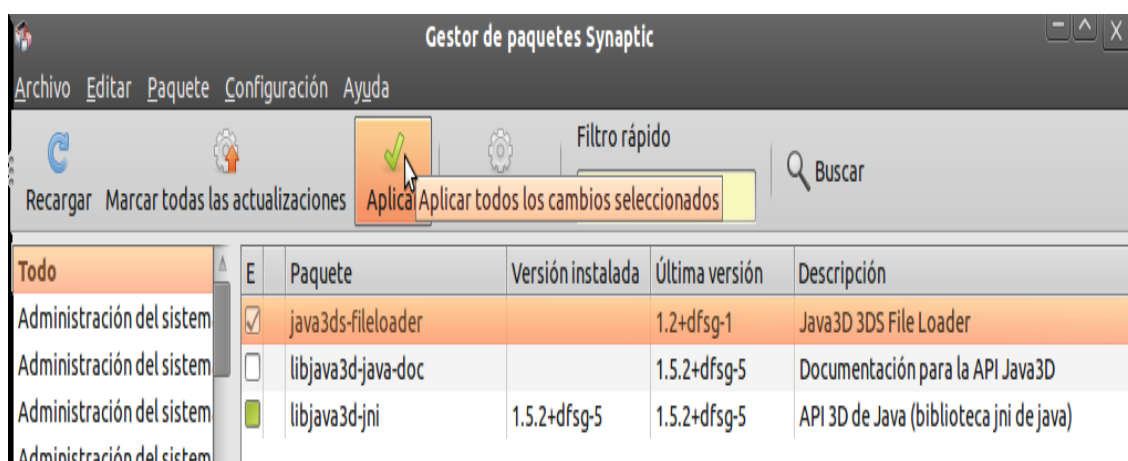


Figura B - 10 Ventana de instalación de Java 3DS File Loader

Fuente: Autores

5.2.1.5 Instalación de Eclipse para CentOS 5

Descargar Eclipse de Internet [81] y extraer el archivo en un directorio. Para este caso en “/opt”:

```
cp -r /root/eclipse-SDK-3.7.2-linux-gtz.tar.gz /usr/local/opt/
tar -xvzf eclipse-SDK-3.7.2-linux-gtk.tar.gz -C /usr/local/opt
```

Como usuario *root* dar permisos de lectura y escritura a todos los archivos en el directorio de Eclipse:

```
chmod -R +r /usr/local/opt/eclipse
```

Crear un archivo ejecutable de Eclipse en la ruta `/usr/bin` y cambiar los permisos:

```
touch /usr/bin/eclipse  
chmod 755 /usr/bin/eclipse
```

Editar el archivo con nano eclipse:

```
nano /usr/bin/eclipse
```

Agregar el siguiente código al archivo en blanco:

```
#!/bin/sh  
export ECLIPSE_HOME="/usr/local/opt/eclipse"  
$ECLIPSE_HOME/eclipse $*
```

Este código únicamente establece la ruta de la variable de `ECLIPSE_HOME` al ejecutar Eclipse.

En terminal colocar “eclipse” con lo que abrirá esta herramienta.

Otra alternativa es dar clic derecho en la barra de menú de aplicaciones en la parte superior de la pantalla, elegir “Editar los menús”, en la sección de “programación”, clic en “Nuevo elemento”, dirigirse al directorio “/usr/bin”, seleccionar el archivo “eclipse”.

Reiniciar el equipo para que tome las variables que fueron editadas.

Para colocar Java 3D en CentOS seguir los pasos de Anexo Configuración Java 3D - API 1.4.0_01 como indica la Figura B - 11

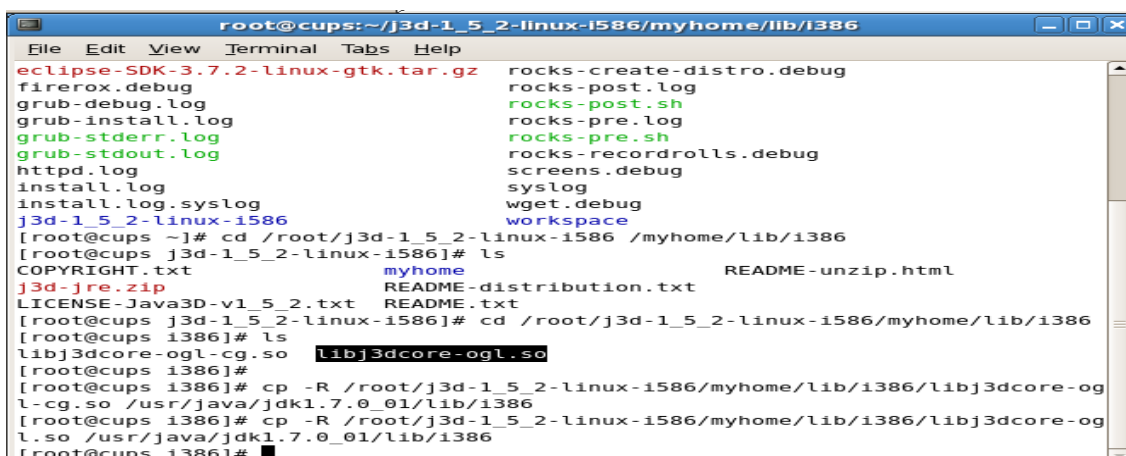


Figura B - 11 Ventana de instalación de Java 3DS File Loader en CentOS

Fuente: Autores

5.2.1.6 Importación de librerías para IDE Eclipse

Una vez seguidos todos los pasos necesarios para la configuración del entorno 3D de Java se procederá a abrir Eclipse, para lo cual hay que dirigirse a Aplicaciones, seleccionar Programas donde ya se podrá visualizar el Eclipse; clic para abrirlo.

Se desplegará una ventana la cual preguntará dónde se guardará el aplicativo a ejecutar, para lo cual se le da una dirección y aceptar:

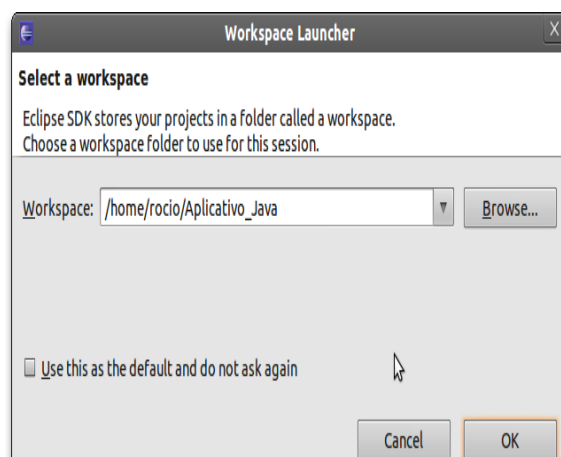


Figura B - 12 Ventana para ubicación del archivo

Fuente: Autores

En la parte superior izquierda, buscar la opción “File” “New”. Para generar un nuevo proyecto seleccionar *New Projects*, se presentará una ventana en la parte izquierda con el nombre del proyecto, dar clic izquierdo en *New* y seleccionar *class* para generar una clase y poder ejecutar el aplicativo. Luego

se procede a agregar las librerías 3D seleccionando “*Properties*” - “*Java Build Path*” - “*Libraries*”:

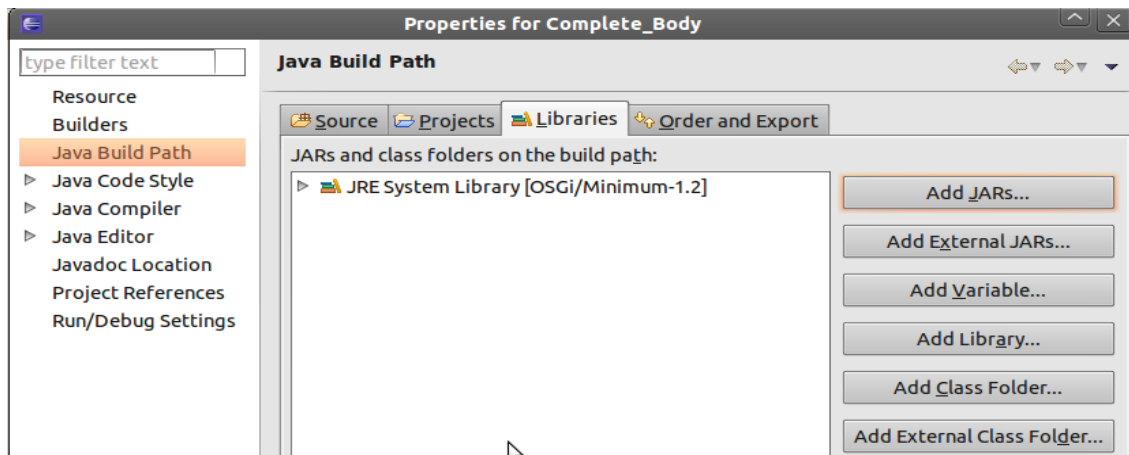


Figura B - 13 Ventana para añadir Librerías

Fuente: Autores

Seleccionar “Add JARs”, buscar la ubicación en la que se encuentran los tres archivos: ‘j3dcore.jar’, ‘j3dutils.jar’, ‘j3d vecmath.jar’:

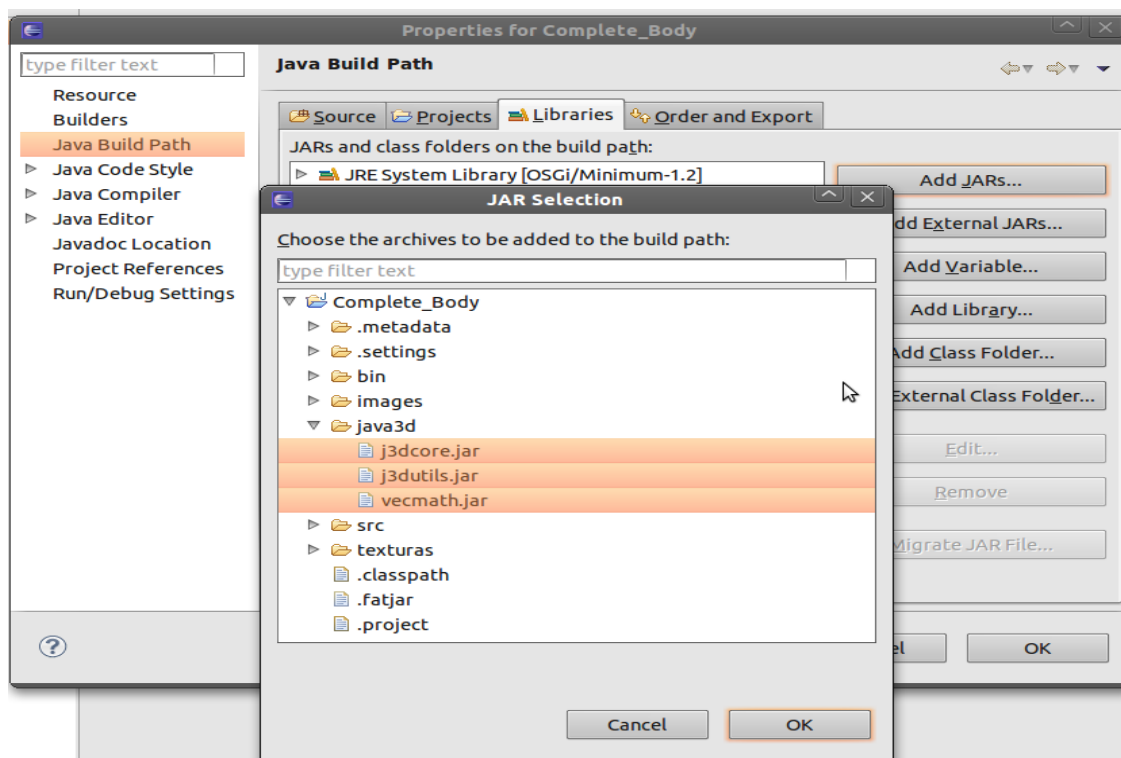


Figura B - 14 Ventana para añadir librerías “.jar”

Fuente: Autores

Dar clic en “OK” y verificar que los archivos se añadieron a la librería

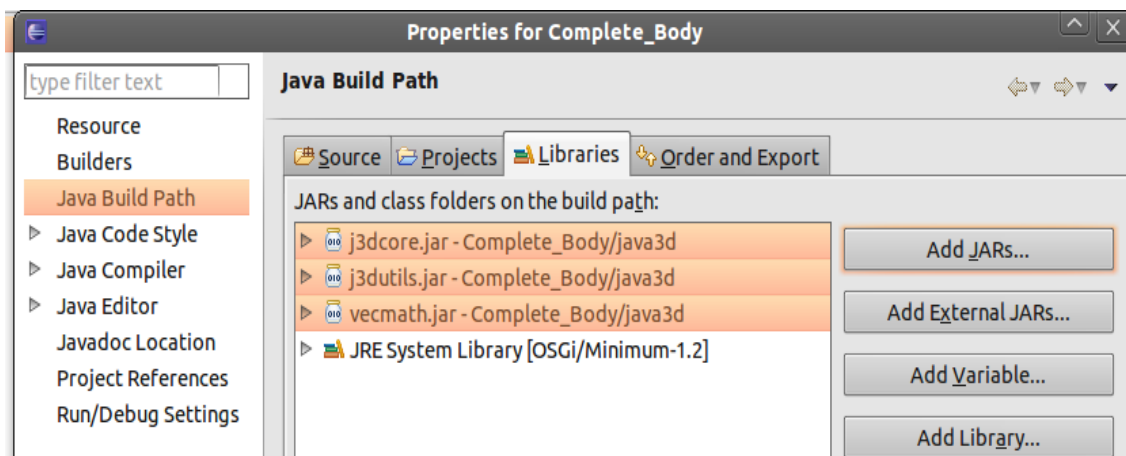


Figura B - 15 Ventana de librerías “.jar” agregadas

Fuente: Autores

Una vez hecho esto se procede a escribir un archivo de prueba como un hola mundo en Java 3D (Ver Anexo Código de verificación Java 3D).

También se pueden importar los proyectos generados para lo cual se creará un nuevo proyecto. Con clic izquierdo seleccionar importar, en la ventana que se despliega escoger “general” y elegir cuál va a ser el método para importar el archivo. Escoger la manera cómo va a ser importado el archivo -> “*Existing Projects into workspace*”, y por último dar clic en “*Next*”

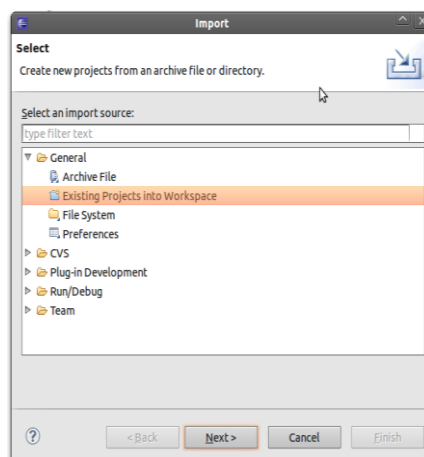


Figura B - 16 Método para importar el proyecto

Fuente: Autores

En la ventana que se despliega, localizar y seleccionar el archivo a extraer.

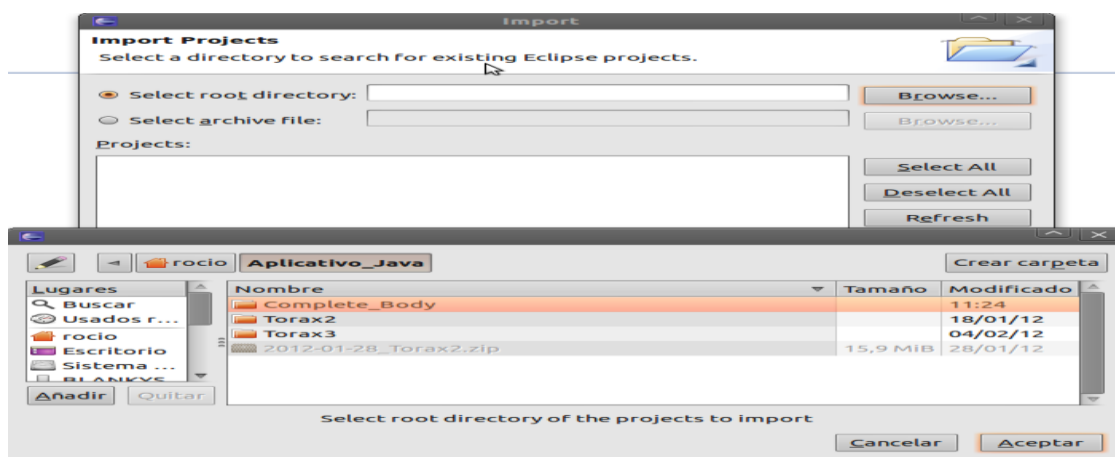


Figura B - 17 Ventana de búsqueda del archivo seleccionado

Fuente: Autores

Después seleccionar todos los archivos para que sean importados y dar clic en finalizar.

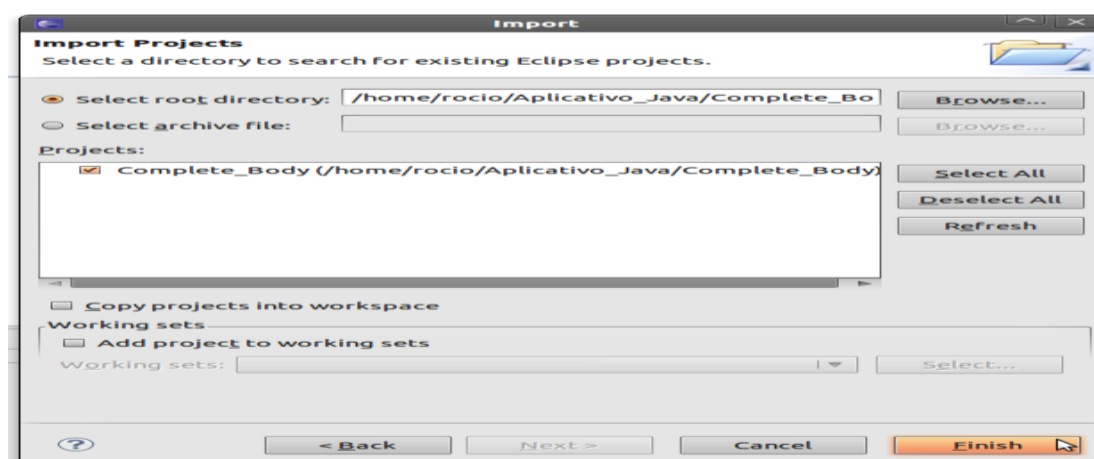



Figura B - 18 Selección de los archivo a importar

Fuente: Autores

Al terminar aparecerán estos archivos en la parte izquierda de la ventana de Eclipse y se puede seguir trabajando en el aplicativo 3D.

5.2.1.7 Código de verificación Java 3D

Para verificar que Java 3D está correctamente instalado y bien direccionado a la ruta o Path descritos en la sección anterior se copiará y colocará el código en Eclipse con el siguiente ejemplo extraído de Internet[82] que muestra el Applet de un cubo en 3D realizado en Java.

Dar clic en el icono  que se encuentra en la parte superior para ejecutar el Applet.

En la Figura 3-6 Applet de Hola Mundo en Java 3Dse puede apreciar la interfaz ejecutada.

5.2.1.8 Compilar y cambiar argumentos en Eclipse

Eclipse por defecto está configurado a un tamaño en memoria de 256Mb o menos para la ejecución de sus programas.

Cuando se trabaja con objetos e imágenes de gran tamaño (2048x2048) y de capacidad en disco grande más de 15Mb, se requiere de un procesador más potente para su ejecución, además de aumentar el tamaño en memoria RAM.

Para aumentar el tamaño en Eclipse dirigirse a “*run/ configuration*” y en la sección de “*Arguments*” escribir lo siguiente: “-Xmx1024m”

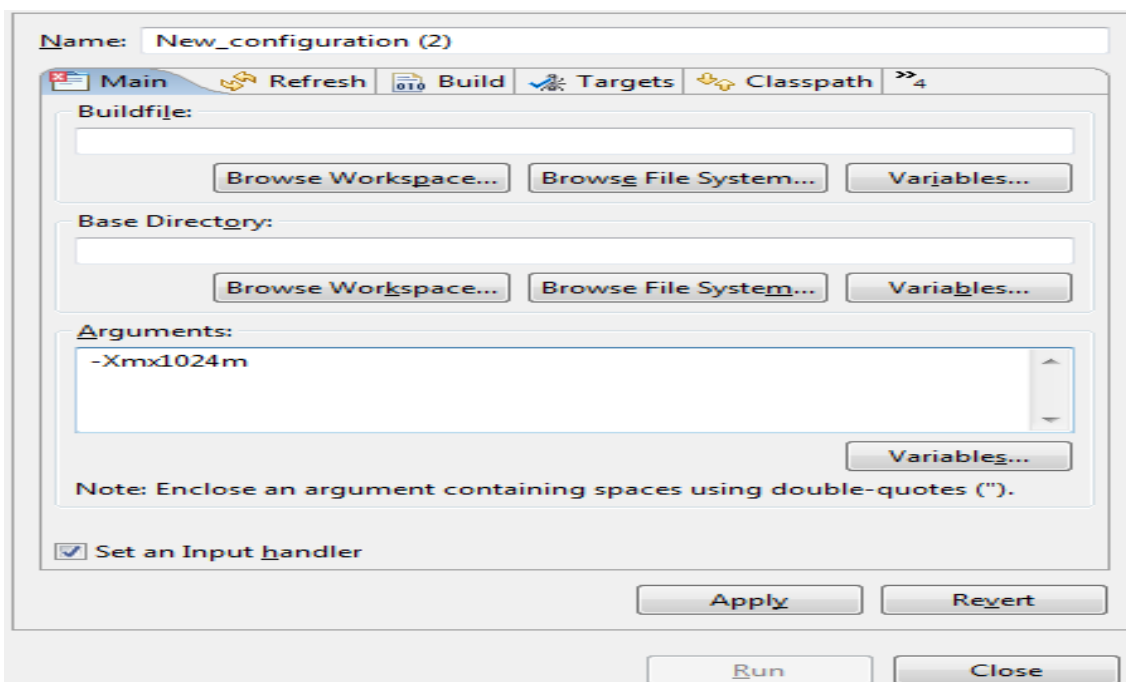


Figura B - 19 Cambio de Argumento en Eclipse

Fuente: Autores

5.2.1.9 Diagrama de clases para el aplicativo 3D

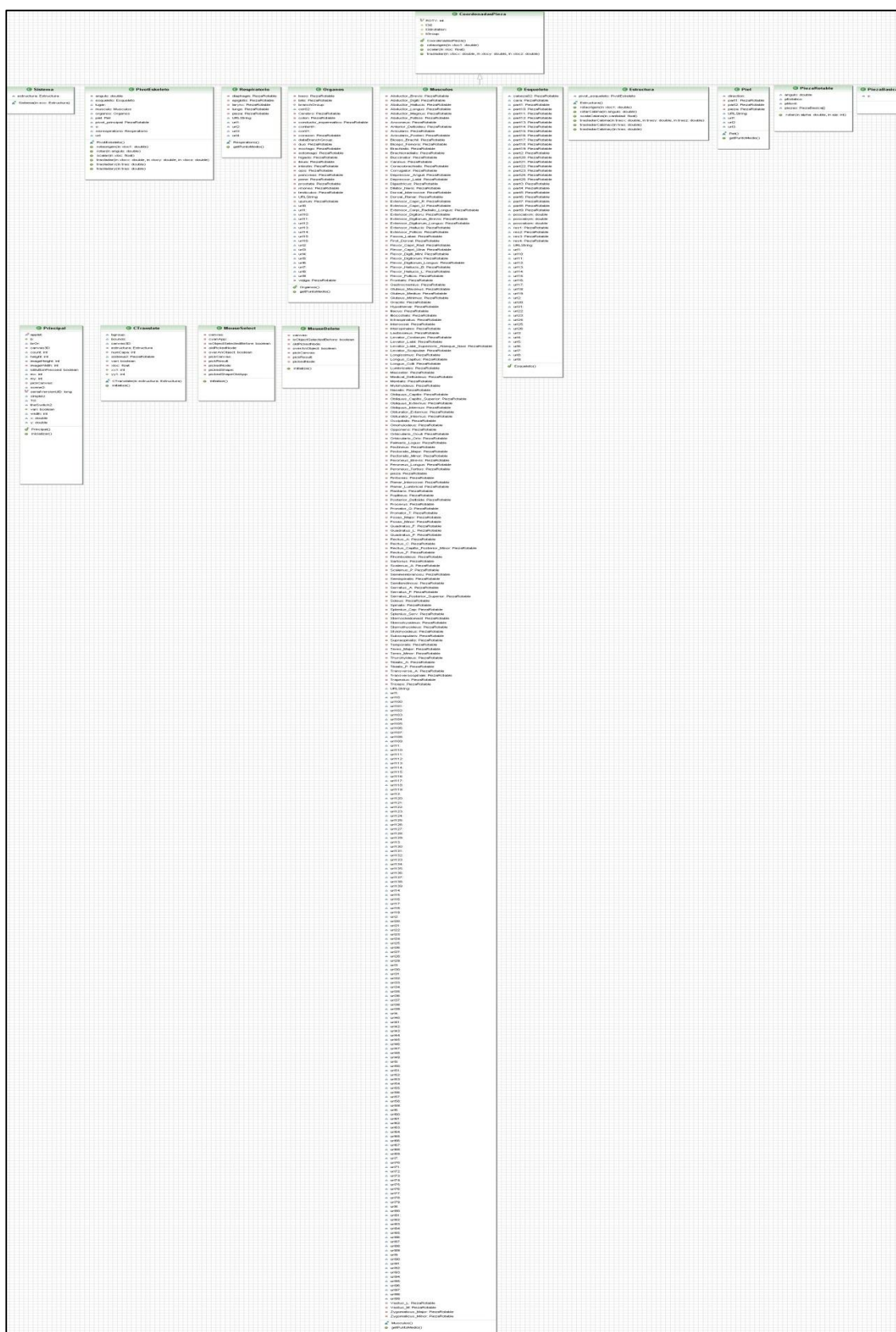


Figura B - 20 Diagrama General de clases Java 3D

Fuente: Los autores

5.3 ANEXO C – HU3: CREACIÓN DE UNA PÁGINA WEB PARA MANIPULACIÓN DEL APPLET DESARROLLADO

5.3.1 Visualización del Applet con HTML

5.3.1.1 Diseño CSS

HTML está limitado a la hora de aplicar forma a un documento, CSS son hojas de estilo en cascada, permite un diseño con estilos, color, técnicas tales como la utilización de tablas, imágenes transparentes para ajustarlas y utilización de etiquetas que no son estándares del HTML.

5.3.1.2 Diseño y Código

- Estilo definido para una etiqueta:

La etiqueta sirve para mostrar un estilo determinado, por ejemplo, se define un párrafo entero en color rojo y otro en color azul. Para ello se utiliza el atributo “style”, que es admitido por todas las etiquetas del HTML (siempre y cuando se disponga de un navegador compatible con CSS):

```
<p style="color:#990000">
Esto es un párrafo de color rojo. </p>
<p style="color:#000099">
Esto es un párrafo de color azul. </p>
```

- Estilo definido en una parte de la página:

Con la etiqueta <DIV> se puede definir secciones de una página y aplicarle estilos con el atributo “style”, es decir, definir estilos de una vez a todo un bloque de la página.

```
<div style="color:#000099; font-weight:bold">
<h3>Estas etiquetas van en <i>azul y negrita</i></h3><p>
Seguimos dentro del DIV, luego permanecen los estilos </p>
</div>
```

- HTML(Style) y CSS

La cabeza de la página HTML está delimitada dentro de la etiqueta<head> y </head>, se debe implementar el diseño con CSS colocando el cuerpo dentro de la etiqueta <STYLE> y </STYLE>, en el cual se colocará el nombre de la etiqueta que se quiere definir los estilos y entre llaves -{}- se colocará la sintaxis CSS con las características de los mismos:

```
<html>
<head>
<title>Ejemplo de estilos para toda una p&aacute;gina</title>
<STYLE type="text/css">
<!--
H1 {text-decoration: underline; text-align:center}
P {font-Family:arial,verdana; color: white; background-color: black}
BODY {color:black;background-color: #cccccc; text-indent:1cm}
// -->
</STYLE>
</head>
<body>
<h1>P&aacute;gina con estilos</h1>
Bienvenidos...
<p>Esto es un ejemplo </p>
</body>
</html>
```

5.4 ANEXO D – HU4: MANIPULACIÓN DEL DISPOSITIVO ELECTRÓNICO

5.4.1 Configuraciones C++

5.4.1.1 Librerías y paquetes de instalación C++

Para compilar el código realizado para la manipulación del dispositivo electrónico en lenguaje C++ se procede a instalar Mingw32 ejecutando el siguiente comando en Ubuntu 11.04:

```
# apt-get install mingw32
```

El comando para la instalación de las librerías básicas Libserial para el manejo del dispositivo electrónico mediante USB es:

```
$ sudo apt-get install -y libserial0 libserial-dev
```

Las librerías para la manipulación de ventanas y botones en C++ con su control de dispositivos de entrada salida es X11 y la lista de paquetes gráficos a instalar desde el Centro de Software de Ubuntu – Gestor de paquetes synaptic son:

```
Paquetes GLUT  
Paquetes X11  
Paquetes xlib  
Paquetes xtail – ext  
Paquetes ruby  
Paquetes Motif  
Paquetes xtst
```

Otra manera de instalación de los paquetes para el uso del dispositivo electrónico en Ubuntu 11.04 es colocando en un archivo en blanco y guardando con “.sh” la siguiente lista de configuración rápida, para posteriormente dar doble clic en el archivo y empezar con la instalación automática:

```
#!/bin/bash
sudo dpkg --configure -a
sudo apt-get install -y subversion
sudo apt-get install -y build-essential
sudo apt-get install -y eclipse
sudo apt-get install -y gcc
sudo apt-get install -y libstdc++6-dev
sudo apt-get install -y libc6-dev
sudo apt-get install -y gcc4.2
sudo apt-get install -y g++
sudo apt-get install -y libgcc1
sudo apt-get install -y libglu1-mesa-dev x11proto-core-dev x11proto-gl-dev
x11proto-fonts x11proto-kb-dev libfontconfig-dev libxpm-dev libxft-dev
sudo apt-get install -y x11-session-utils x11-xfs-utils x11-common x11-xkb-utils x11-
apps x11-xserver-utils x11-utils
sudo apt-get install -y libx11-xcb1 libx11-xcb-dev libx11-xcb-dbg libhugs-x11-
bundled libghc6-x11-dev libxcb1 libxcb1-dev
sudo apt-get install -y xtail libinput-pad-xtest libxcb-xtest0-dbg libxcb-xtest0 libxcb-
xtest0-dev libxext6 libxext-dev libxext6-dbg
sudo apt-get install -y libmixlib-authentication-ruby1.8 libmixlib-cli-ruby1.8 libmixlib-
config-ruby1.8
sudo apt-get install -y libmotif-dev libmotif4-dbg libmotif4
sudo apt-get install -y pxlib-dev pxlib1 libxtst-dev
sudo apt-get install -y libserial0 libserial-dev
```

Una vez instalados todos los paquetes necesarios para utilizar la librería libserial ya se puede trabajar en Eclipse, donde se creará un nuevo proyecto y en “properties/settings/libraries” añadir el nombre “*serial*”.

Otra manera de instalar la librería serial para compilar el código en CentOS es descargarse el libserial-0.5.0.tar.gz [83], descomprimirlo y dentro de la carpeta tipear lo siguiente en un terminal como superusuario:

```
./configure  
make  
make install
```

Para ejecutar el código realizado para la manipulación del dispositivo electrónico en C++ desde un terminal se procede a tipearlo siguiente:

```
g++ -lserial -lXtst perifericos.cpp perifericos.h usbport.cpp usbport.h main.cpp -o  
dispositivoc
```

Donde se incluirán: la librería *lserial* con *lXtst* seguida de los archivos .cpp y librerías .h realizadas, junto con el nombre del ejecutable a generar como: *dispositivoc.exe*

Dar permisos de lectura/escritura al puerto USB para lo cual se requiere ingresar como superusuario y tipear lo siguiente:

```
chmod a+rw /dev/ttyUSB0
```

Indicar la dirección de la librería serial para utilizarla

```
export LD_LIBRARY_PATH=/usr/local/lib/:$LD_LIBRARY_PATH
```

Ejecuta el archivo del guante

```
./dispositivoc
```

5.4.1.2 Diagrama de clases para el aplicativo con el guante electrónico de datos

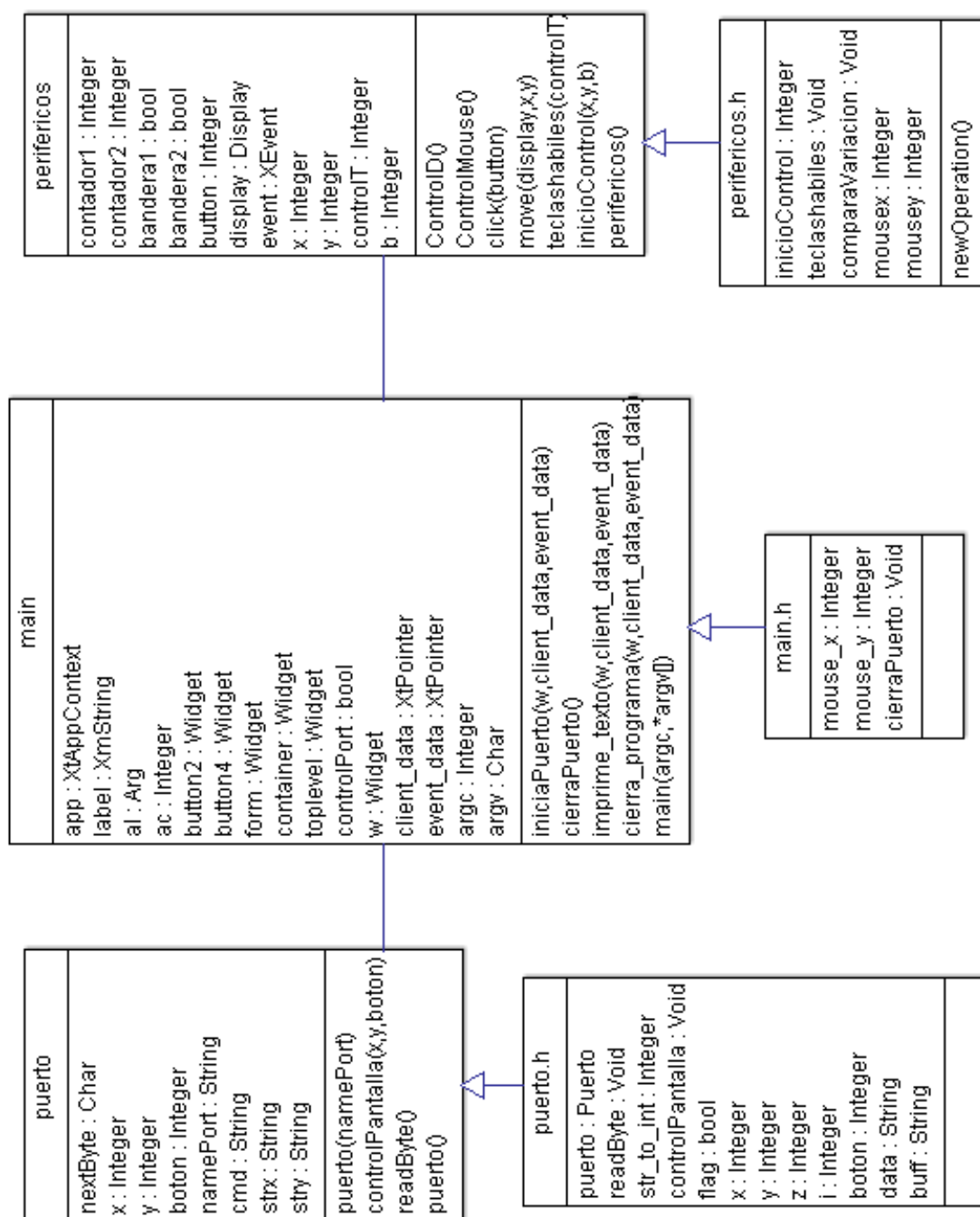


Figura D - 1 Diagrama General de clases Java 3D

5.5 ANEXO E – HU5: APLICATIVO DESARROLLADO INCLUIDO EN EL CLÚSTER

5.5.1 Configuración e Instalación de Rocks Clúster

5.5.1.1 Comandos para trabajar con el Clúster

Algunos comandos útiles para manipular el Clúster se detallan a continuación:

Comandos para manipulación del Clúster	
Comando	Explicación
scp prog.sh root@cups:	Para transferir archivos desde el computador hacia srvcuerpo
scp -r /home/migi/Documentos/UPS_3D user3d@190.15.136.10:/home/user3d	Si la copia es de un directorio usar la opción -r
scp prog.sh migi@migi:	Para copiar un archivo del clúster hacia el computador
scp -r midir migi@migi:	Para copiar un directorio usar la opción -r
Hostname cat /etc/hosts cat /etc/sysconfig/network-scripts/ifcfg-eth0 cat /etc/sysconfig/network-scripts/ifcfg-eth1 cat /usr/global/sge-6.2u5-bin/default/common/act_master	Para mostrar las direcciones IP, local host, nombre_máquina, cat /etc/hosts
rocks run host compute ps	Para comprobar los procesos activos en todos los nodos compute de un clúster
ssh compute-0-3 df	Para ejecutar un comando sobre un nodo mediante SSH
ps -u ps -aux	Para ver que procesos están ejecutándose con ps con opciones como: -aux Lista los procesos de todos los usuarios con información añadida -a Lista los procesos de todos los usuarios

Comandos para manipulación del Clúster	
Comando	Explicación
	<p>-u Muestra el usuario al que pertenece el proceso y la hora de inicio, la utilización de Cpu y memoria, etc.</p> <p>-x Muestra los procesos que no están controlados por ningún terminal.</p> <p>-l Muestra información que incluye el UID y el valor "nice".</p> <p>-forest Muestra el listado procesos en un formato tipo árbol que permite ver como los procesos interactúan entre sí, Combinadas para una visión global de los procesos.</p>
kill -TERM <PID>	Se informa al proceso (hay que especificar el PID del proceso, no el nombre) que queremos que termine
kill -KILL <PID>	Cuando un proceso está <i>zombie</i> , colgado o no responde, se usa este para eliminarlo directamente de la cola de ejecución.
killall<nombre proceso>.	Si se quiere matar un proceso de forma más cómoda
skill -STOP -u <nombre usuario>	Si se quiere detener todas las ejecuciones de un determinado usuario
CTRL+ C pausarlo con CTRL+ Z devolverlo a primer plano con fg	Cuando se está ejecutando un programa en consola y se quiere terminar, se envía directamente la señal TERM.
top -c:	Para visualizar la línea de comandos completa de cada proceso, activado mostrará las rutas completas, mientras que desactivando solo muestra el nombre del programa
top -d	Intervalo de actualización y refresco, se puede asignar un valor numérico (segundos) que determinará cada cuanto actualice la información
top -U	Monitorizar los procesos de un determinado UID
top -p	Monitoriza los ID de procesos especificados
top -n	Se especifica el nº de veces que actualizará hasta que finalice la ejecución de Top. Por ejemplo: -n4#,refrescará la información cuatro veces y finalizará la ejecución de TOP.
ssh root@compute-0-0	Conexión al clúster con el cliente ssh de Linux

Comandos para manipulación del Clúster	
Comando	Explicación
ssh root@192.168.3.2	
ssh -XY root@cluster.dominio	Conexión con el cliente ssh de Linux usando X11 - forward
ssh -x root@cluster.dominio	Conexión con el cliente ssh de Linux sin usar X11 - forward
passwd	Comando para cambio de contraseña en Linux
ping -c 2 compute-0-0.local	Verificación de la conectividad con uno de los nodos del clúster
ping -b 10.0.0.0 -c 2 compute-0-0.local	Verificación de la conectividad con todos los nodos del clúster
touch archivo-vacío-frontend	Creación de un archivo en blanco
\$ ssh c0-4 w	Ejecución remota en uno de los nodos del comando w para ver los usuarios loggeados en el nodo
cluster-fork -n "c0-2 c0-3 c0-5" w	Ejecución de un comando en varios nodos del clúster, escogiendo los nodos
cluster-fork -n "c0-0 c0-1" "ps r -A -o user,%cpu,%	Muestra todos los procesos en ejecución en algunos nodos del clúster como: el usuario, el % de CPU utilizado, el % de RAM, el tiempo que lleva en ejecución y el nombre sintético del comando.
ls -ld/export	Muestra las propiedades de enlace simbólico / export
export - v	Muestra la lista de los sistemas de archivos compilados vía NFS
ls -l /export/	Revisa el contenido del directorio /export
service httpd status	Muestra el estado del servicio httpd
service httpd start	Inicia el servicio httpd
service 411 commit	Sincroniza los archivos de configuración que han cambiado recientemente
make -C /var/441	Igual que el anterior
make -C /var/441 force	Sincroniza TODOS los archivos de configuración
ls -al /export/home/usuario	Muestra el contenido (incluso el oculto) del directorio casa de usuario
useradd usuario	Crea una cuenta de usuario
usermod -d /export/home/usuario usuario	Cambia el directorio casa del usuario en los archivos de configuración
passwd usuario	Fija la contraseña de usuario
ssh usuario@localhost	El usuario se conecta como usuario al frontend
rocks sync user	Finaliza la creación de cuentas de usuario(Se

Comandos para manipulación del Clúster	
Comando	Explicación
	ejecuta después de useradd y passwd)
usermod -g <gid> usuario	Cambia el número de grupo de un usuario
userdel usuario	Borra la cuenta de usuario
umount /home/usuario	Desmonta el home directory de usuario
cluster-fork umount /home/usuario	Desmonta el home directory de usuario en todo el clúster
rm -rf /export/home/usuario	Borra el home directory en el frontend

Tabla E - 1 Comandos usados para manipular el Clúster

Fuente: Autores

5.5.1.2 Configuración de librerías gráficas

Verificar que las librerías gráficas estén disponibles colocando la sentencia “glxinfo” en un Terminal. Si se despliega un error en la librería “Xlib” referente a la tarjeta gráfica, corregir colocando lo siguiente:

```
/sbin/init 3
```

El sistema cambia de ejecución gráfica a ejecución en modo consola, en la pantalla a desplegarse introducir usuario y contraseña para continuar. Colocar los siguientes comandos:

```
service nvidia start
nvidia-xconfig
/sbin/init 6
```

Para reiniciar el sistema si no lo ha hecho tipear:

```
init 6
```

Revisar que el navegador Web este corriendo desde la red interna de la Universidad Politécnica Salesiana ingresando “190.15.136.20/ganglia”

Si aparece el siguiente error:

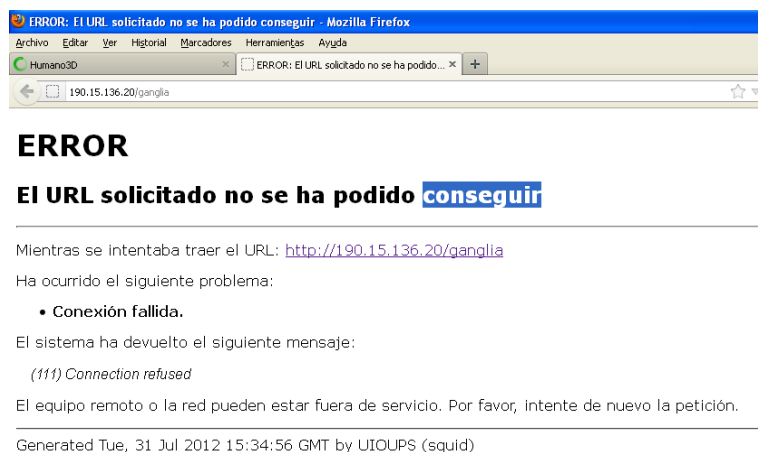


Figura E - 1 Error de conexión

Fuente: Autores

Este error es debido a las seguridades en el servidor proxy para solucionarlo colocar en el Rocks en modo de Superusuario, lo siguiente:

```
service iptables stop
```

Ingresar nuevamente a Ganglia y verificar el acceso. Posteriormente permitirá ingresar a la página Web del cuerpo humano sin inconvenientes, para habilitar nuevamente el servidor proxy colocar:

```
service iptables start
```

5.5.1.3 Instalación y configuración de los Nodos

“VSphere Client” será la herramienta utilizada para la creación y utilización de un Clúster virtual.

Antes de la instalación del Clúster virtual se procede a crear un espacio de memoria de la siguiente manera:

Clic en 172.17.40.100 / Configuration /Networking para conocer qué red (VMNetwork) se puede conectar con los Nodos.

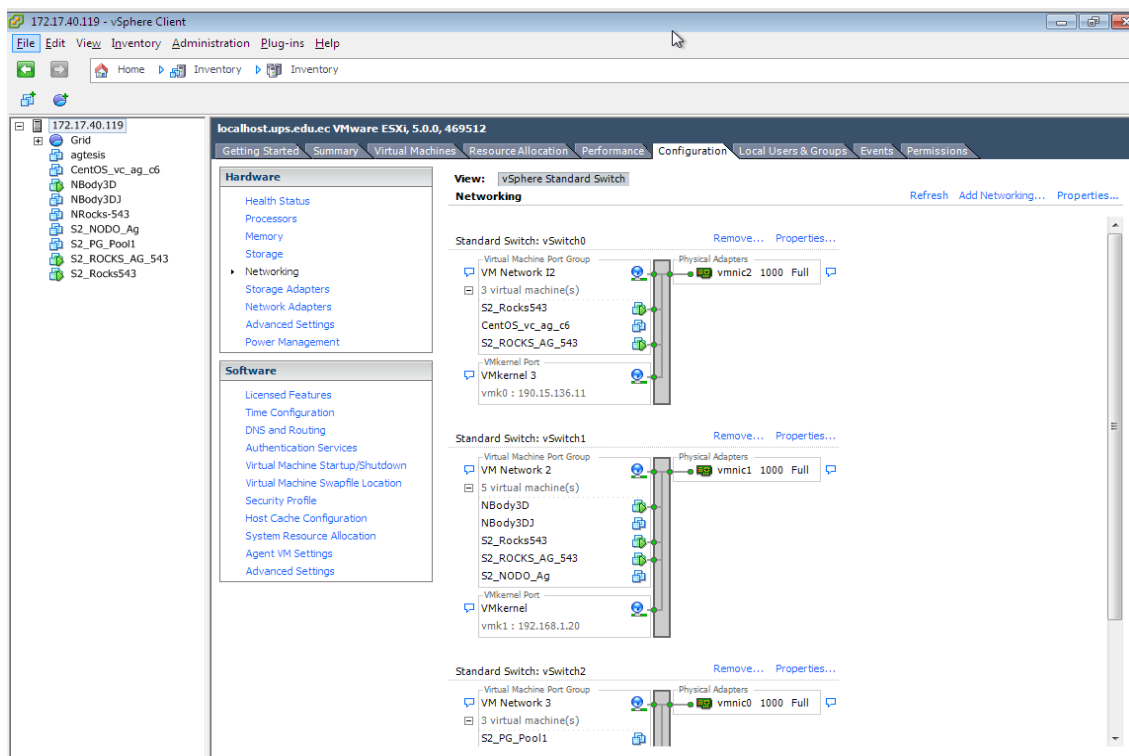


Figura E - 2 VMware

Fuente: Autores

A continuación se describen los pasos para crear el Clúster virtual:

Clic derecho en New Virtual Machine/Typical-next/ proporcionar un nombre con el cual se conocerá al Clúster“R2_CuerpoHumano”. Clic en next hasta finalizar.

Seleccionar el sistema operativo – Linux y la versión - CentOS 4/5/6 (64bit) / “Next”.

Colocar la red “VM Network 2” para la red avanzada.

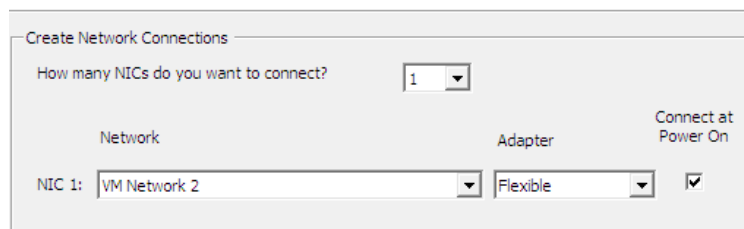


Figura E - 3 Nueva Conexión VMware

Fuente: Autores

Elegir el tamaño del disco virtual – 60GB /next/finish

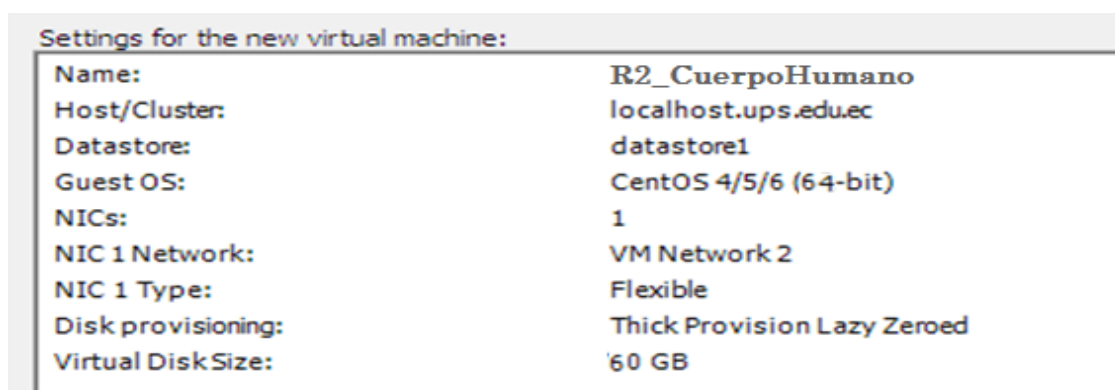


Figura E - 4 Tamaño de disco VMware

Fuente: Autores

En la parte izquierda de la ventana de la máquina virtual aparecerá el Clúster generado, clic derecho sobre este y elegir “edit settings”, en la pantalla a desplegarse seleccionar CD/DVD drive 1 para que lo reconozca e iniciar la instalación:

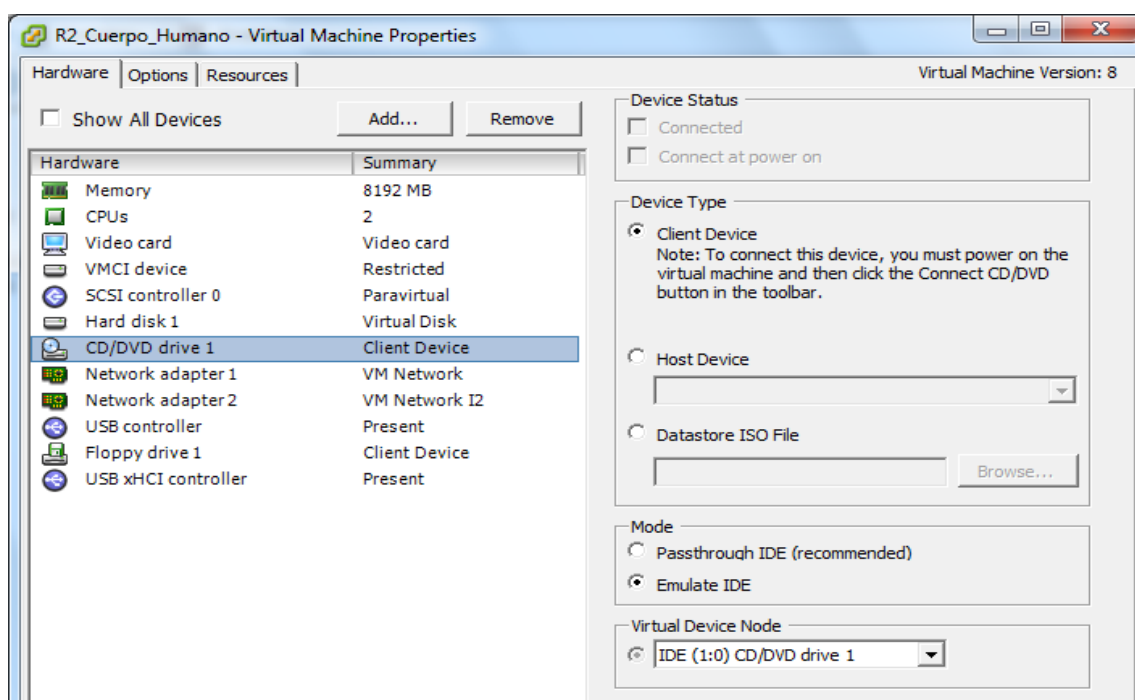


Figura E - 5 Ventana de edit settings

Fuente: Autores

Finalizada la asignación del espacio de memoria se procede a la instalación del Clúster como se indicó en la Iteración 1. Configuración de herramientas para el uso y paralelización de la aplicación 3D, para lo cual seleccionar “Connect at power on” y “Host Device” para que reconozca el CD en el que se encuentra el instalador llamado “área 51”.

Para iniciar con la instalación del nodo físico el cual debe contener la tarjeta gráfica Nvidia, es necesario dirigirse al FrontEnd e iniciar el proceso colocando en un terminal “*insert-ethers*”, en la ventana que aparece, elegir la opción “*tile*” y “ok”, con lo cual detectará automáticamente al nodo físico a ser instalado(ver Figura E - 6).

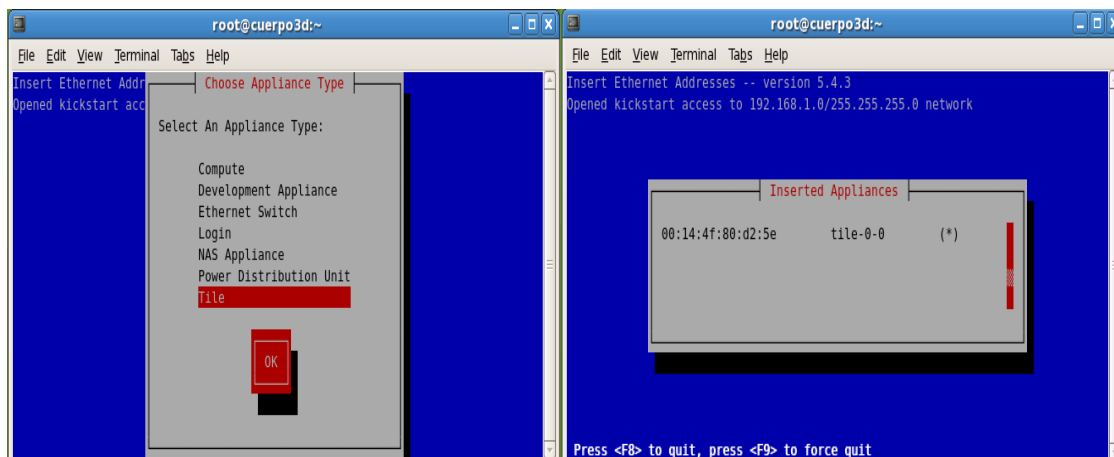


Figura E - 6 Selección de opción “tile”

Fuente: Autores

Después de ejecutar el comando anterior hay que encender el nodo físico con el CD de “Kernel Roll” para comenzar a instalar.

5.5.1.4 Configuración final de los nodos

Si la instalación de los nodos es exitosa no queda más que comenzar a utilizar el Clúster. Pero se puede hacer una revisión previa para saber si todo funciona:

- Un usuario puede conectarse exitosamente al FrontEnd
- Una vez dentro verificar si se puede conectar transparentemente a los nodos.

5.5.1.5 Eliminación de cuentas de usuario

Para ello se usa el comando `userdel` y `-r` para eliminar todas las sub carpetas:

```
# userdel -r user3d
```

- Desmontado del home directory:

```
#umount /home/user3d
```

- Eliminación del home directory:

```
#rm -rf /export/home/user3d
```

- Sincronización de los archivos:

```
rocks sync users
```

- Eliminar la entrada en:

```
"/etc/auto.home"
```

5.5.1.6 Instalación de Cmake

Cmake es una plataforma de código abierto siendo una herramienta diseñada para crear y probar paquetes de software. Se utiliza para controlar el proceso de compilación de software usando una plataforma sencilla y archivos de configuración independientes del compilador *m*, este genera makefiles nativos y áreas de trabajo que se pueden utilizar en el entorno del compilador de su elección.

Para instalar buil-essential desde CentOS tipear en consola como user3d:

```
# yum groupinstall "Development Tools"
# yum install kernel-devel kernel-headers
```

Para instalar los repositorios en CentOS, descargar el archivo rpm de internet³⁰ y ejecutarlo dando doble clic sobre este.

La instalación de cmake se la puede realizar de dos maneras:

- Tipear en un terminal de FrontEnd lo siguiente:

```
yum install cmake
```

- O descargar el paquete CMAKE de Internet³¹ y colocarlo dentro de root con el respectivo permiso:

```
cd /usr/local/src
```

³⁰Descargar de: http://packages.sw.be/rpmforge-release/rpmforge-release-0.5.2-2.el5.rf.x86_64.rpm

³¹Descargar de: <http://www.cmake.org/files/v2.6/cmake-2.6.1.tar.gz>

Luego se procederá con estos pasos:

```
tar xzvf cmake-2.6.1.tar.gz ; cd cmake-2.6.1
yum install ncurses-devel
./bootstrap --prefix=/usr/local/cmake-2.6.1
make
make install
ln -s /usr/local/cmake-2.6.1 /usr/local/cmake
export PATH=/usr/local/cmake/bin:$PATH
export MANPATH=/usr/local/cmake/man:$MANPATH
```

5.5.2 Aplicativo con OpenSceneGraph

5.5.2.1 Instalación de paquetes

Para la instalación de OSG son necesarios algunos requerimientos, comprobar desde el “Gestor de paquetes Synaptic” si se encuentran configurados, caso contrario proceder con la instalación individual como muestra la Figura E - 7:

- g++
- freeglut3
- libc6 (>= 2.4)
- libgcc1 (>= 1:4.1.1)
- libgl1-mesa-glx | libgl1
- libglu1-mesa | libglu1
- libopenscenegraph65 (>= 2.8.3)
- libopenthreads13 (>= 2.8.3)
- libstdc++6 (>= 4.5)
- libsdl1.2-dev
- libsdl-mixer1.2-dev
- Libtiff (3.4-1)
- Libxml2 (2.7.6)
- Libxml2 ++1.0c2a
- Curl (7.17.1)
- FFmpeg

- Qt libraries 4.6.1

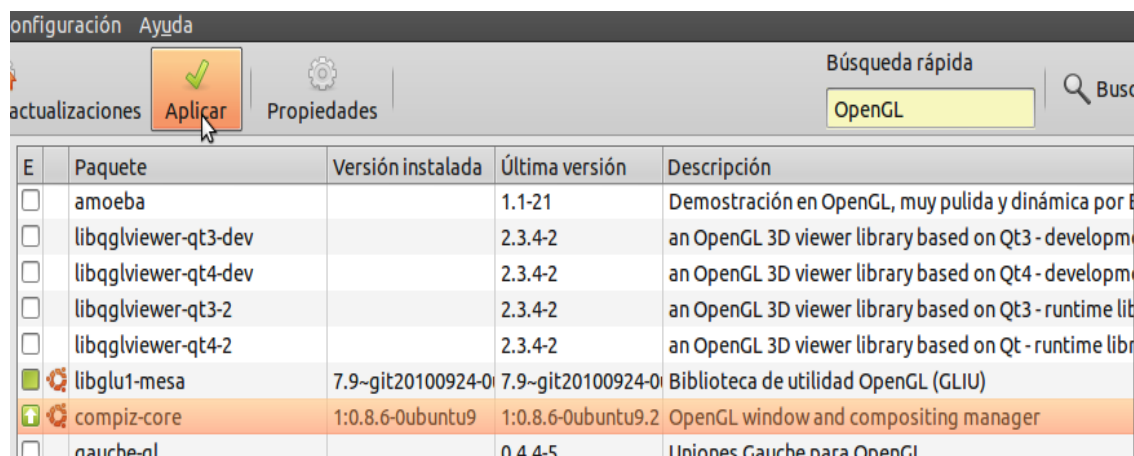


Figura E - 7 Ventana de librería a comprobar

Fuente: Autores

Para comprobar si están instaladas las librerías de C++ abrir un terminal y colocar la siguiente sentencia:

```
g++ --version
```

Para la ejecución de gráficos en Ubuntu 11.04 es preciso tener instalados todos los paquetes necesarios. Para comprobar tipear el siguiente comando.

```
glxinfo
```

Si despliega el siguiente error:

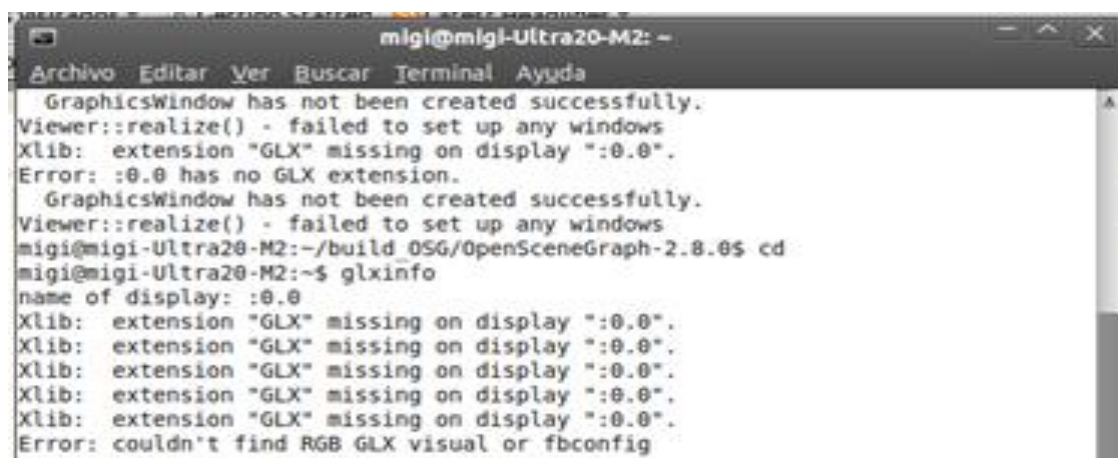


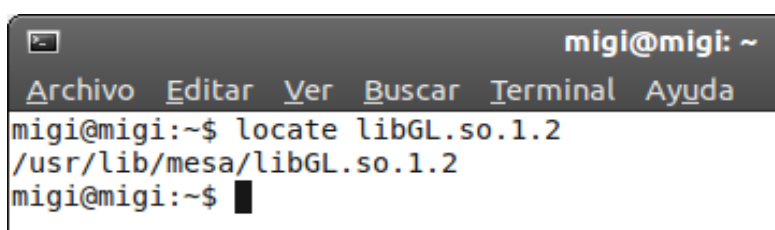
Figura E - 8 Ventana de error de librería gráfica

Fuente: Autores

Mandar a localizar donde está ubicado el archivo “libGL.so.1.2” con el siguiente comando:

```
locate libGL.so.1.2
```

Si la librería mesa está instalada, saldrá la ruta donde se ubica el archivo:



```
migi@migi: ~
Archivo Editar Ver Buscar Terminal Ayuda
migi@migi:~$ locate libGL.so.1.2
/usr/lib/mesa/libGL.so.1.2
migi@migi:~$
```

Figura E - 9 Ventana de librería libGL.so.1.2 a comprobar localización

Fuente: Autores

Dirigirse a “Sistema/Administración/Gestor de Paquetes Synaptic” y proceder a desinstalar “nvidia-current”:



Figura E - 10 Ventana de librería Nvidia a desinstalar

Fuente: Autores

Hecho esto reiniciar el equipo para que los cambios se ejecuten.

Salvo el caso que no se pueda aún visualizar que se ha incorporado la parte gráfica y siga dando el mismo error desinstalar todos los archivos de “Nvidia” buscando en “Synaptic”, reiniciar y volver a colocar en un terminal el comando “glxinfo” para lo cual ya no debe desplegar ningún error.

5.5.2.2 Instalación de OSG

Obtener el código fuente de openscenegraph.org y extraer los archivos:

OpenSceneGraph-3.0.1.zip

OpenSceneGraph-Data-3.0.0.zip³²

Extraer los archivos en la carpeta “/home/user/”. Allí se tiene todos los códigos fuente en directorio “/home”.

Instalar dependencias desde Ubuntu 11.04:

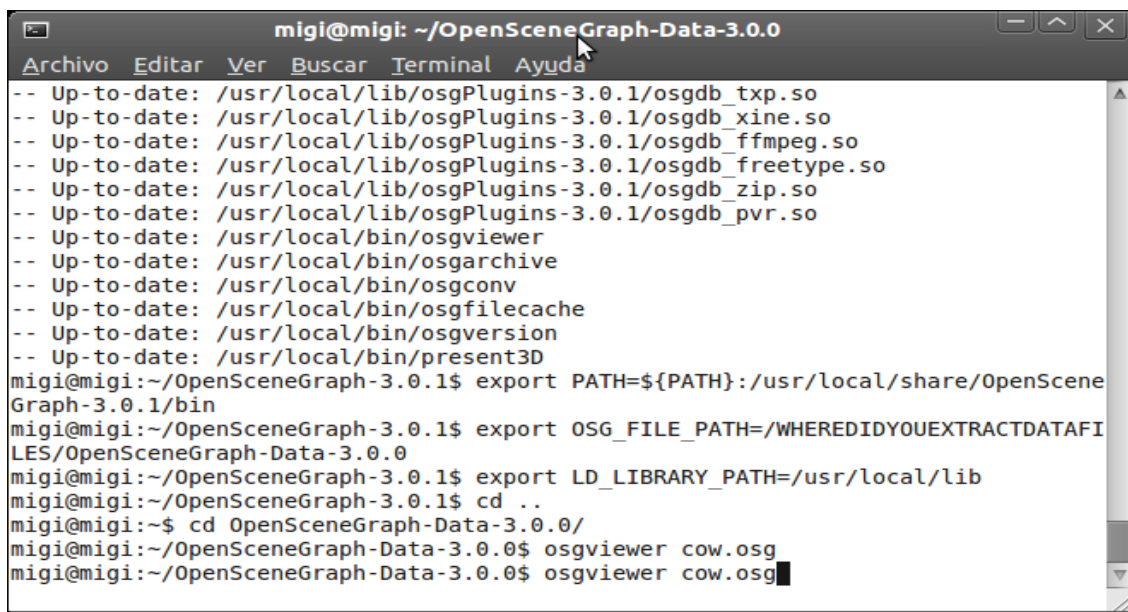
```
sudo apt-get build-dep openscenegraph
sudo apt-get install cmake
```

Para instalar dependencias y *Cmake* en CentOS ver anexo Instalación de Cmake

Continuar con los siguientes comandos escritos en un Terminal para la instalación y configuración de OSG como muestra la Figura E - 11:

```
cd OpenSceneGraph-3.0.1/
./configure
make
sudo make install
cmake ../OpenSceneGraph-3.0.1 -DCMAKE_BUILD_TYPE=Release
make
sudo make install
export PATH=${PATH}:/usr/local/share/OpenSceneGraph-3.0.1/bin
export
OSG_FILE_PATH=/WHERE DID YOU EXTRACT DATA FILES/OpenSceneGraph-
Data-3.0.0
export LD_LIBRARY_PATH=/usr/local/lib
cd ..
cd OpenSceneGraph-Data-3.0.0/
osgviewer cow.osg
```

³²Descargar de: <http://www.openscenegraph.com/index.php/downloads/data>



```

migi@migi: ~/OpenSceneGraph-Data-3.0.0
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_txp.so
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_xine.so
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_ffmpeg.so
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_freetype.so
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_zip.so
-- Up-to-date: /usr/local/lib/osgPlugins-3.0.1/osgdb_pvr.so
-- Up-to-date: /usr/local/bin/osgviewer
-- Up-to-date: /usr/local/bin/osgarchive
-- Up-to-date: /usr/local/bin/osgconv
-- Up-to-date: /usr/local/bin/osgfilecache
-- Up-to-date: /usr/local/bin/osgversion
-- Up-to-date: /usr/local/bin/present3D
migi@migi:~/OpenSceneGraph-3.0.1$ export PATH=${PATH}:/usr/local/share/OpenScene
Graph-3.0.1/bin
migi@migi:~/OpenSceneGraph-3.0.1$ export OSG_FILE_PATH=/WHERE DID YOU EXTRACT DATA FI
LES/OpenSceneGraph-Data-3.0.0
migi@migi:~/OpenSceneGraph-3.0.1$ export LD_LIBRARY_PATH=/usr/local/lib
migi@migi:~/OpenSceneGraph-3.0.1$ cd ..
migi@migi:~$ cd OpenSceneGraph-Data-3.0.0/
migi@migi:~/OpenSceneGraph-Data-3.0.0$ osgviewer cow.osg
migi@migi:~/OpenSceneGraph-Data-3.0.0$ osgviewer cow.osg

```

Figura E - 11 Ventana de instalación de OpenSceneGraph

Fuente: Autores

Luego de colocado en el Terminal “osgviewer cow.osg” esperar un momento y aparecerá la Figura3-7 Ventana de compilación del “Hola Mundo” con OSG:

Luego de instalado y ejecutado el “Hola mundo” de OSG con el “cow.osg” ejecutar en un terminal

Idconfig

Que permitirá asegurarse de que está actualizado con las últimas librerías instaladas de OSG.

5.5.2.3 Código de ejemplo y librerías de OSG

Las librerías necesarias para la ejecución de OpenSceneGraph son:

```

#include<osgViewer/Viewer>// librería de visor de OSG
#include<osg/Node>// librería de Nodos
#include<osg/PositionAttitudeTransform>//librería de posiciones, rotaciones,
#include<osg/Geode>//uso de figuras geométricas
#include<osg/ShapeDrawable>// librería para graficar Shapes
#include<osg/Drawable> // uso de gráficos en pantalla
#include<osg/Material>//uso de materials y texturas

```

```

#include<osg/Shape>// librería para utilizar Shapes
#include<osg/StateSet>// librería para Escenas
#include<osg/Camera>// manejo de Cámaras
#include<osg/ref_ptr>// para referir a librerías de OSG
#include<osgDB/Registry>// registro de archivos
#include<osgDB/WriteFile>// escritura de archivos
#include<osgDB/ReadFile>// lectura de archivos
#include<osg/Notify> // notificaciones y errores
#include<iostream>// librería de C++
#include<osg/Geometry>// creación de Geometrías
#include<osg/Array> // Arreglos en OSG
#include<osg/Group>// agrupa Nodos y los muestra
#include<osgGA/GUIEventHandler>// Eventos de Entrada/Salida
#include<fstream>// Ficheros creación y lectura

```

La creación de una mano mediante figuras geométricas de tipo cilindros (verFigura E - 12 Creación de una mano con OSG utilizando geometrías.) y rotación con Método “Quat” propio de OSG, se muestra en el siguiente código:

```

// Rotación de los dedos - demo
usingnamespace osg;
intmain(int argc, char **argv ) {
    osgViewer::Viewer* viewer=new osgViewer::Viewer();//Creando el Viewer
    osg::ref_ptr<osg::Group> root = new osg::Group;//Creando el nodo root
    //Código para la creación de un cilindro //El objeto Geode contiene la figura
    ref_ptr<Geode> cylGeode = newGeode;ref_ptr<Geode> esfGeode = newGeode;
    ref_ptr<Geode> esfGeodePULGAR00 = newGeode;
    ref_ptr<Geode> esfGeodeINDICE00 = newGeode;
    ////Posiciona las imágenes en el lugar que se especifique.
    PositionAttitudeTransform* dedopulg0X=newPositionAttitudeTransform();
    PositionAttitudeTransform* dedoindi0X=newPositionAttitudeTransform();
    Vec3 dedo1X(0,0,0);// varia la posición de cada
    Vec3 dedo2X(0,-6,5);

```

```

dedopulg0X ->setPosition(dedo1X); // varía solo en posición - rota todos
dedoindi0X ->setPosition(Vec3(0,-6,5));
//// código para reacción de la mano  //// dedo pulgar 1
osg::ShapeDrawable *dedopulg01 = new    osg::ShapeDrawable(new
osg::Capsule(Vec3(0,0,-2),1.3,2));
    dedopulg01->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f));// blanco //RGBA
esfGeode->addDrawable(dedopulg01);
    osg::ShapeDrawable *dedopulg02 = new    osg::ShapeDrawable(new
osg::Capsule(Vec3(0,0,1),1.3,2));
    dedopulg02->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f));// blanco //RGBA
esfGeodePULGAR00->addDrawable(dedopulg02);
dedopulg0X ->setAttitude(Quat(-1.0,0.0,0.0,1.0));// X
    dedopulg0X ->addChild(esfGeodePULGAR00);
root->addChild(dedopulg0X);
//// Dedo Indice 1
    osg::ShapeDrawable *dedoindi01 = newShapeDrawable(new
Capsule(Vec3(3,0,0),1,2.7));
dedoindi01->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f)); // seteo de Color RGBA
esfGeode->addDrawable(dedoindi01);
    osg::ShapeDrawable *dedoindi02 = newShapeDrawable(new
Capsule(Vec3(3,0,3.5),1,2.7));
dedoindi02->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f)); // seteo de color RGBA
esfGeode->addDrawable(dedoindi02);
    osg::ShapeDrawable *dedoindi03 = newShapeDrawable(new
Capsule(Vec3(3,0,7),1,2.7));
    dedoindi03->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f));
    esfGeodeINDICE00->addDrawable(dedoindi03);
dedoindi0X ->setAttitude(Quat(-1.0,0.0,0.0,1.0));// X
    dedoindi0X ->addChild(esfGeodeINDICE00);
root->addChild(dedoindi0X);
// Palma de la mano 001
osg::ShapeDrawable *palma01 = new    osg::ShapeDrawable(new
osg::Capsule(Vec3(6,0,-8),8,2));

```

```
palma01->setColor(osg::Vec4f(1.0f, 0.5f, 0.1f, 1.0f)); // blanco
esfGeode->addDrawable(palma01);
root->addChild(esfGeode);
    viewer->setUpViewInWindow(500,100,600,500);
    viewer->setSceneData(root);
    viewer->run();
}
```

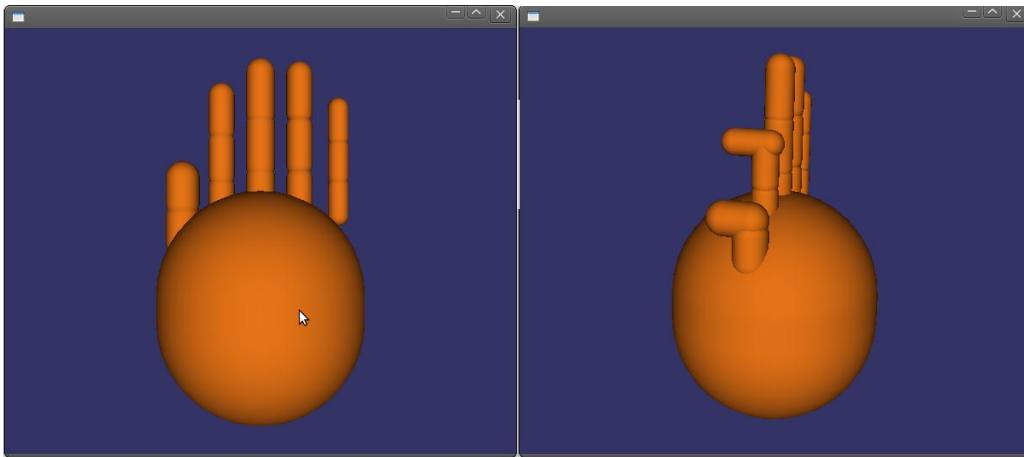


Figura E - 12 Creación de una mano con OSG utilizando geometrías.

Fuente: Autores

5.5.2.4 Diagrama de clases para el aplicativo 3D realizado con la librería OSG

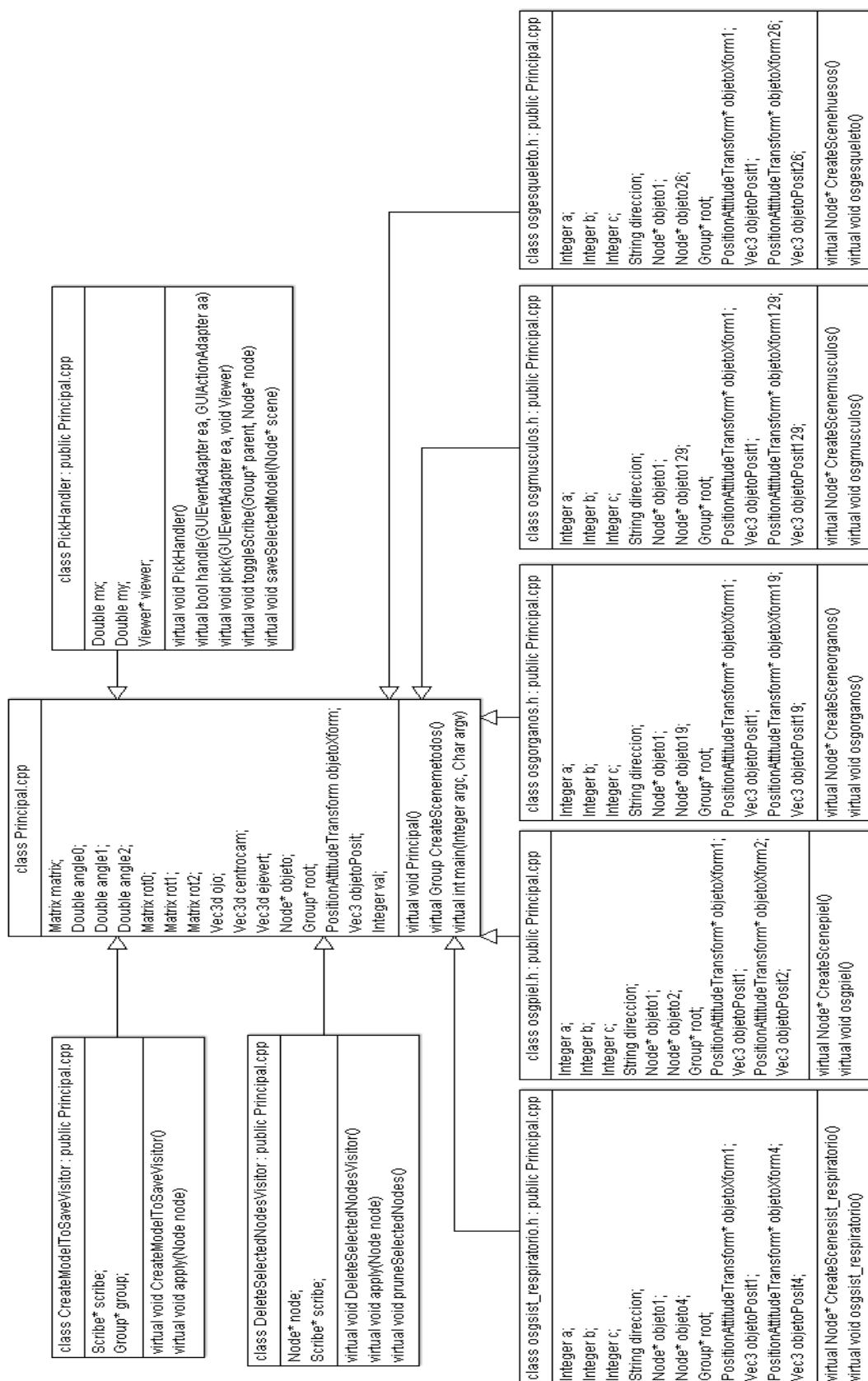


Figura E - 13 Diagrama de Clases OSG.

Fuente: Los autores

5.5.2.5 Proceso de compilación con MPI

El proceso de compilación difiere entre las distintas implantaciones de MPI.

Generalmente, el comando para compilar incluye todas las banderas del compilador necesarias para incluir tanto los archivos cabecera como las bibliotecas.

```
mpiCC -losgWidget -losgViewer -losgText -losgSim -losgParticle -losgGA -losgDB -
losg -losgVolume -losgUtil -losgTerrain -losgShadow -losgManipulator -losgFX -
losgAnimation -lOpenThreads Principal.cpp -o cuerpohu.exe
```

MPI es independiente de la implementación y puede necesitar varios scripts, argumentos y variables de ambiente, por lo que para el proceso de ejecución se procede a colocar en un archivo en blanco y guardando con “.sh” lo siguiente para el aplicativo con OSG:

```
export PATH=/usr/lib64/openmpi/1.4-gcc/bin:$PATH
export LD_LIBRARY_PATH=/usr/lib64/openmpi/1.4-gcc/lib:$LD_LIBRARY_PATH
mpirun -np 1 -machinefile listanodos.txt
/home/user3d/OSG_3D/osgbody/cuerpohu.exe
```

5.5.2.6 Instalación de paquetes en el nodo

Para que el aplicativo CGLX ejecute tanto en el FrontEnd como en los nodos de visualización es requerida la instalación de *freeglut*.

Extraer el archivo *freeglut.2.8.0.tar.gz* en la carpeta “/home/user3d/”, ya que allí se tiene todos los códigos fuente. Continuar con los siguientes comandos escritos en un terminal para la instalación y configuración del archivo en el FrontEnd:

```
# tar xvzf /home/user3d/freeglut.2.8.0.tar.gz
# cd freeglut.2.8.0
# ./autogen.sh
# ./configure
```

```
# su
$ make
$ make install
```

Para la instalación y configuración en los nodos se copiará el archivo *freeglut* dentro del “/home/user3d/” de los nodos, luego ingresar a estos por ssh como superusuario y asignar una contraseña al root.

```
scp -r /home/user3d/freeglut-2.8.0 tile-0-1:/home/user3d
# ssh tile-0-1
# passwd root
cuerpo123
```

Entrar como usuario a los nodos, al archivo *freeglut* dar permisos de ejecución e ingresar a este y comenzar la ejecución como se indica a continuación:

```
# su user3d
$ cd /home/user3d/
$ chmod -R 777 freeglut-2.8.0/
$ cd freeglut-2.8.0/
$ ./configure
$ make
$ su
# make install
# cd src/
# make
# make install
# cd ..
$ cd include/
$ make
$ su
# cd include/
# make install
```

```
$ cd GL/
$ make
$su
# cd GL
# make install
```

Para verificar que la instalación tuvo éxito dirigirse a la carpeta de librerías y confirmar que los archivos están allí:

```
# cd /usr/local/lib
# ls -all
```

5.5.2.7 Proceso de compilación con CGLX

El proceso de compilación difiere de las anteriores ya que para ejecutar el documento se tendrá que generar un archivo con extensión *.pro* como muestra la Figura E - 14 donde se colocará información requerida como: el nombre del archivo a generar “*mybody0*”, el lenguaje a compilar, la dirección de las librerías.

```
TARGET      = mybody0
PROJECT      = TUTORIAL
VERSION      = 1
TEMPLATE     = app
LANGUAGE     = C
CONFIG       += warn_off release
CONFIG       -= qt
DEFINES      += DEF_USE_CGLX
# -----
# Configuración común
# -----
DESTDIR      = ./
OBJECTS_DIR  = .obj
HEADERS      += *.h
SOURCES      += *.cpp
```

```

# linux
# -g++
# -----
linux-g++ {
    LIBS      += /usr/local/lib/libglm.a -L/usr/local/lib -
L/opt/HIPerWorks/Pireen/include -L/opt/HIPerWorks/Pireen/include/pirCore -
L/usr/lib64 -L/opt/HIPerWorks/Pireen/lib64 -lpirCore -lglut -lGLU -lGL -lpng -ljpeg -lz
    INCLUDEPATH += /usr/include/GL /opt/HIPerWorks/Pireen/include
    # clean it
    CLEAN_FILES = ./${DESTDIR}${TARGET}
}
# linux
# -g++-64
# -----
linux-g++-64 {
    LIBS      += /usr/local/lib/libglm.a -L/usr/local/lib -
L/opt/HIPerWorks/Pireen/include -L/opt/HIPerWorks/Pireen/include/pirCore -
L/usr/lib64 -L/opt/HIPerWorks/Pireen/lib64 -lpirCore -lglut -lGLU -lGL -lpng -ljpeg -lz
    INCLUDEPATH += /usr/include/GL /opt/HIPerWorks/Pireen/include
    # clean it
    CLEAN_FILES = ./${DESTDIR}${TARGET}
}
linux-gcc-64 {
    LIBS      += /usr/local/lib/libglm.a -L/usr/local/lib -
L/opt/HIPerWorks/Pireen/include -L/opt/HIPerWorks/Pireen/include/pirCore -
L/usr/lib64 -L/opt/HIPerWorks/Pireen/lib64 -lpirCore -lglut -lGLU -lGL -lpng -ljpeg -lz
    INCLUDEPATH += /usr/include/GL /opt/HIPerWorks/Pireen/include
    # clean it
    CLEAN_FILES = ./${DESTDIR}${TARGET}
}
# -----
CLEAN_FILES += /*~ ${OBJECTS_DIR}/* ./Makefile

```

Figura E - 14 Archivo .pro

Fuente : Autores

En un terminal colocar las siguientes sentencias para compilar el archivo *.pro*:

```
make clean
qmake
make
```

Luego compilar el archivo CGLX y tipear en un terminal con el nombre del archivo ejecutable que se creó como:

```
./mybody0
```

Si se genera un error mencionando que falta el archivo "*libglm.la*" en el directorio "*glm*", se debe copiar este del registro original glm-0.3.1.tar.gz ubicado en la carpeta "*glm*" dentro del proyecto que se está compilando.

Si se forma otro error de no encontrar el archivo o directorio libglut.so.3 colocar la siguiente sentencia en un terminal:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

Para compilar el programa CGLX con la herramienta HiperWorks y los nodos de visualización para la parelización visual; en los nodos ingresar a la carpeta en el que se encuentre el archivo y tipear la dirección de la librería *freeglut* previamente instalada, ejecutar el demonio y luego en el usuario seleccionar el aplicativo CGLX como indica la Figura 3-74 y ejecutar.

```
cd /home/user3d/bodycglx_originalOK/
//colocar el export en las maquinas en las que se va a ejecutar el aplicativo
export LD_LIBRARY_PATH=/usr/local/lib
//en los tile
pirdaemon
```


GLOSARIO

A

API: Application programming interface. pag. 22

Applet: Es un componente de software que corre en el contexto de otro programa, por ejemplo un navegador web. pag. 17

AutoCAD: Auto hace referencia a la empresa creadora del software, Autodesk y CAD a Diseño Asistido por Computadora pag 45

B

Byte: es la unidad fundamental de datos en los ordenadores personales, un byte son ocho bits contiguos. - pag. 24

C

CentOS: Community Enterprise Operating System. Sistema operativo de la plataforma Linux – pag. 52

CGI: Interface de Acceso Común. Estos son programas usados para hacer llamadas a rutinas o bien para controlar otros programas como también bases de datos desde una página Web. – pag 16

Clúster: un conjunto de computadoras interconectadas con dispositivos de alta velocidad que actúan en conjunto usando el poder de cómputo de varias CPUs en combinación para resolver ciertos problemas dados - pag. 1.

F

FrontEnd: es responsable de recoger entradas de los usuarios, y ser procesadas de tal manera que cumplan las especificaciones para que se pueda usarlas - pag 12

G

Gedit: aplicación para edición de archivos de texto. pag. 144

J

Java Runtime Environment (JRE): Utilizado para ejecutar programas en Java, está compuesto por una máquina virtual Java la cual es capaz de ejecutar el bytecode y las librerías estándar de Java. pag. 67

Java Development Kit (JDK): Siendo su antiguo nombre SDK, constituye un conjunto de herramientas, utilidades, documentación y ejemplos para el desarrollo de aplicaciones Java. pag. 34

Java Virtual Machine (JVM): La JVM forma parte del entorno de ejecución de Java (JRE) y lo que ésta interpreta son ByteCodes que son generados al compilar cualquier programa Java, siendo idénticos para todas las plataformas. pag. 320

JPEG: Joint Photographic Experts Groupe, Unión de Grupo de Expertos Fotográfico. Formato gráfico con compresión con pérdidas que consigue elevados ratios de compresión - pag.46

L

Linux: es el nombre abreviado de GNU/Linux, un Sistema Operativo libre, hecho por hackers, compatible con UNIX y disponible en Internet para su descarga gratuita y legal. - pag. 12

M

Mapping: es el método de adición de detalles, superficie o colores a un modelo 2D o 3D generado por computadora - pag. 74

O

Open Graphics Library (OpenGL): es una interfaz de software para el hardware de gráficos - pag. 26

OSCAR(Open Source Clúster Application Resources) es un paquete de software designado a simplificar la instalación de clúster - pag 50

P

Pixel: Picture Element, es un único punto en una imagen gráfica. Los monitores gráficos muestran imágenes dividiendo la pantalla en miles (o millones) de píxeles, dispuestos en filas y columnas. Los píxeles están tan juntos que parece que estén conectados. - pag. 57

Plugin: es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande. - pag 4

R

Root: Es el nombre de la cuenta de usuario que posee todos los derechos en todos los modos – pag 12

Roll: Paquetes de software adicionales que permiten ampliar el sistema mediante la integración sin problemas y de forma automática en los mecanismos de gestión y embalaje utilizados por el software de base- pag 50

S

Superusuario: Conocido también como root, usuario administrador de Linux – pag 12

T

Toolkit: Conjunto de componentes y herramientas implementados en diferentes aplicaciones, utilizando versiones para plataformas nativas de los componentes. - pag. 28

W

Wavefront: es un formato de archivo de definición de la geometría desarrollada primero por “Wavefront Technologies” para su paquete de animación avanzada – pag 7

Visualizer. El formato de archivo es abierto y ha sido adoptado por otros proveedores de aplicaciones gráficas 3D - pag. 69

Wine Is Not an Emulator (Wine): un software para instalar aplicaciones nativas de Windows en Linux, de forma sencilla. pag. 76

Wireless: es un término usado para describir las telecomunicaciones en las cuales las ondas electromagnéticas (en vez de cables) llevan la señal sobre parte o toda la trayectoria de la comunicación. - pag. 2